

# 9.1 Индексация

## Шаг 1

### Тема урока: строки

1. Индексация строк
2. Итерирование строк
3. Решение задач

**Аннотация.** Строковый тип данных. Вспомним основные операции над строками, научимся работать с отдельными символами, а также перебирать (итерировать) символы строк.

### Повторение материала

Строки в Python используются, когда надо работать с текстовыми данными.

**Создание строки.** Для создания строк мы используем парные кавычки '' или "" :

```
s1 = 'Python'  
s2 = "Pascal"
```

**Считывание строки.** Для считывания текстовых данных в строковую переменную мы используем функцию input() :

```
s = input() # считали текст  
num = int(input()) # считали текст и преобразовали его в целое число
```

**Пустая строка.** Для создания пустой строки мы пишем s = '' или s = "" . Пустая строка – это аналог числа 0 .

**Длина строки.** Для определения длины строки (количество символов) мы используем встроенную функцию len() :

```
s = 'Hello'  
n = len(s) # значение переменной равно 5  
print(n)
```

**Конкатенация и умножение на число.** Операторы + и \* можно использовать для строк. Оператор + сцепляет две и более строк. Это называется конкатенацией строк. Оператор \* повторяет строку указанное количество раз.

Выражение	Результат
'AB' + 'cd'	'ABcd'
'A' + '7' + 'B'	'A7B'
'Hi'* 4	'HiHiHiHi'

**Оператор принадлежности in .** С помощью оператора in мы можем проверять, находится ли одна строка в составе другой. То есть, является ли одна строка подстрокой другой:

```
s = 'All you need is love'  
if 'love' in s:  
    print('❤')  
else:  
    print('💔')
```

Так как строка s содержит подстроку 'love' , то будет выведен смайлик ❤ .



В Python можно использовать смайлики emoji (<https://pypi.org/project/emoji/>) 😊

## Индексация строк

Очень часто бывает необходимо обратиться к конкретному символу в строке. Для этого в Python используются квадратные скобки `[ ]`, в которых указывается индекс (номер) нужного символа в строке.

Пусть `s = 'Python'`. Таблица ниже показывает, как работает индексация:

Выражение	Результат	Пояснение
<code>s[0]</code>	P	первый символ строки
<code>s[1]</code>	y	второй символ строки
<code>s[2]</code>	t	третий символ строки
<code>s[3]</code>	h	четвертый символ строки
<code>s[4]</code>	o	пятый символ строки
<code>s[5]</code>	n	шестой символ строки



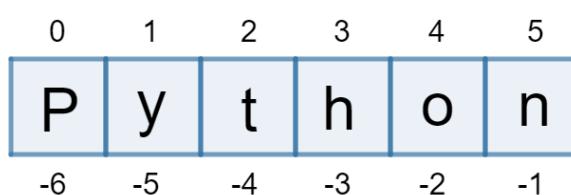
Обратите внимание, первый символ строки равен `s[0]`, а не `s[1]`. В Python индексация начинается с `0`, по аналогии с функцией `range(n)`, которая генерировала последовательность натуральных чисел от `0` до `n - 1`.

В отличие от многих языков программирования, в Python есть возможность работы с отрицательными индексами. Если первый символ строки имеет индекс `0`, то последнему элементу присваивается индекс `-1`.

Выражение	Результат	Пояснение
<code>s[-6]</code>	P	первый символ строки
<code>s[-5]</code>	y	второй символ строки
<code>s[-4]</code>	t	третий символ строки
<code>s[-3]</code>	h	четвертый символ строки
<code>s[-2]</code>	o	пятый символ строки
<code>s[-1]</code>	n	шестой символ строки

Таким образом, получаем

Положительные индексы	0	1	2	3	4	5
Строка	P	y	t	h	o	n
Отрицательные индексы	-6	-5	-4	-3	-2	-1



Частая ошибка у начинающих программистов – обращение по несуществующему индексу в строке.

Например, если `s = 'Python'`, и мы попытаемся обратиться к `s[17]`, то мы получим ошибку:

```
IndexError: string index out of range
```

Ошибка возникает, поскольку строка содержит всего 6 символов.



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)`.

## Итерирование строк

Очень часто нужно просканировать всю строку целиком, обрабатывая каждый ее символ. Для этого удобно использовать цикл `for`. Напишем программу, которая выводит каждый символ строки на отдельной строке:

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

Результатом выполнения такой программы будут строки:

```
a  
b  
c  
d  
e  
f
```

Мы передаем в функцию `range()` длину строки `len(s)`. В нашем случае длина строки `s`, равна 6. Таким образом, вызов функции `range(len(s))` имеет вид `range(6)` и переменная цикла `i` последовательно перебирает все значения от 0 до 5. Это означает, что выражение `s[i]` последовательно вернет все символы строки `s`. Такой способ итерации строки удобен, когда нам нужен не только сам элемент `s[i]`, но и его индекс `i`.

Если нам не нужен индекс самого символа, то мы можем использовать более короткий способ итерации:

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

Этот цикл пройдет по строке `s`, придавая переменной цикла `c` значение каждого символа (!) в отличие от предыдущего цикла, в котором переменная цикла «бегала» по индексам строки.



Обратите внимание на обозначение переменных цикла. В первом цикле мы используем имя `i`, что соответствует стандартной идеологии наименования переменных цикла. Во втором цикле, мы назвали переменную буквой `c` – первая буква слова char (символ).

❤️ Happy Pythoning! 🧑

## Шаг 2

Что покажет приведённый ниже код?

```
s = 'abcdefg'  
print(s[0] + s[2] + s[4] + s[6])
```

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/2>

## Шаг 3

Что покажет приведённый ниже код?

```
s = 'abcdefg'  
print(s[0]*3 + s[-1]*3 + s[3]*2 + s[3]*2)
```

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/3>

## Шаг 4

Что покажет приведённый ниже код?

```
s = '01234567891011121314151617'  
for i in range(0, len(s), 5):  
    print(s[i], end='')
```

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/4>

## Шаг 5

Используя индексатор, дополните приведённый ниже код так, чтобы он вывел символ запятой.

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/5>

## Шаг 6

Используя индексатор, дополните приведённый ниже код так, чтобы он вывел символ `w`.

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/6>

## Шаг 7

### В столбик 1

На вход программе подаётся одна строка. Напишите программу, которая выводит элементы строки с индексами `0, 2, 4, ...` в столбик.

### **Формат входных данных**

На вход программе подаётся одна строка.

### **Формат выходных данных**

Программа должна вывести элементы строки с индексами `0, 2, 4, ...`, каждое на отдельной строке.

► Тестовые данные 

#### **Sample Input:**

```
abcdefghijklmnpqr
```

#### **Sample Output:**

```
a  
c  
e  
g  
i  
k  
m  
o
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/7>**

## **Шаг 8**

### **В столбик 2**

На вход программе подаётся одна строка. Напишите программу, которая выводит в столбик элементы строки в обратном порядке.

### **Формат входных данных**

На вход программе подаётся одна строка.

### **Формат выходных данных**

Программа должна вывести в столбик элементы строки в обратном порядке.

► Тестовые данные 

#### **Sample Input:**

```
abc
```

#### **Sample Output:**

```
c  
b  
a
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/8>**

## Шаг 9

### ФИО

На вход программе подаются три строки: имя, фамилия и отчество (именно в таком порядке). Напишите программу, которая выводит инициалы человека.

#### Формат входных данных

На вход программе подаются три строки, каждая на отдельной строке.

#### Формат выходных данных

Программа должна вывести ФИО человека.

**Примечание.** Гарантируется, что имя, фамилия и отчество начинаются с заглавной буквы.

► Тестовые данные 

#### Sample Input:

```
Тимур  
Гуев  
Ахсарбекович
```

#### Sample Output:

```
ГТА
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/9>**

## Шаг 10

### Цифра 1

На вход программе подаётся одна строка состоящая из цифр. Напишите программу, которая считает сумму цифр данной строки.

#### Формат входных данных

На вход программе подаётся одна строка, состоящая из цифр.

#### Формат выходных данных

Программа должна вывести сумму цифр данной строки.

► Тестовые данные 

#### Sample Input:

```
2514
```

#### Sample Output:

```
12
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/10>**

## Шаг 11

### Цифра 2

На вход программе подаётся одна строка. Напишите программу, которая выводит сообщение «Цифра» (без кавычек), если строка содержит цифру. В противном случае вывести сообщение «Цифр нет» (без кавычек).

#### Формат входных данных

На вход программе подаётся одна строка.

#### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

► Тестовые данные 

#### Sample Input 1:

Hi Python

#### Sample Output 1:

Цифр нет

#### Sample Input 2:

Hi 17 Python

#### Sample Output 2:

Цифра

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/11>

## Шаг 12

### Сколько раз?

На вход программе подаётся одна строка. Напишите программу, которая определяет, сколько раз в строке встречаются символы  и 

Символ + встречается <n> раз  
Символ \* встречается <m> раз

где ,  и 

#### Формат входных данных

На вход программе подаётся одна строка.

#### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

► Тестовые данные 

#### Sample Input:

bcd+a++++\*\*31415

#### Sample Output:

Символ + встречается 5 раз  
Символ \* встречается 2 раз

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/12>

## Шаг 13

### Одинаковые соседи

На вход программе подаётся одна строка. Напишите программу, которая определяет сколько в ней одинаковых соседних символов.

#### Формат входных данных

На вход программе подаётся одна строка.

#### Формат выходных данных

Программа должна вывести количество одинаковых соседних символов.

► Тестовые данные 

**Sample Input 1:**

abcd

**Sample Output 1:**

0

**Sample Input 2:**

aabbcc

**Sample Output 2:**

3

**Sample Input 3:**

aaa

**Sample Output 3:**

2

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/13>

## Шаг 14

### Гласные и согласные

На вход программе подаётся одна строка с буквами русского языка. Напишите программу, которая определяет количество гласных и согласных букв и выводит текст в следующем формате:

Количество гласных букв равно <кол-во гласных букв>

Количество согласных букв равно <кол-во согласных букв>

#### Формат входных данных

На вход программе подаётся одна строка.

## Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание 1.** В русском языке 9 гласных букв ( ауыиэяю ) и 21 согласная буква ( бвгджзйклмнпрстфхцчш ).

**Примечание 2.** Ваша программа должна игнорировать все небуквенные символы, а также букву ё .

► Тестовые данные

### Sample Input 1:

Вдохновение – это умение приводить себя в рабочее состояние!

### Sample Output 1:

Количество гласных букв равно 25

Количество согласных букв равно 24

### Sample Input 2:

Глупая критика не так заметна, как глупая похвала.

### Sample Output 2:

Количество гласных букв равно 18

Количество согласных букв равно 23

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/14>

## Шаг 15

### Decimal to Binary 10

На вход программе подаётся натуральное число, записанное в десятичной системе счисления. Напишите программу, которая переводит данное число в двоичную систему счисления

([https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F\\_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0\\_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)).

## Формат входных данных

На вход программе подаётся одно натуральное число.

## Формат выходных данных

Программа должна вывести число записанное в двоичной системе счисления.

► Тестовые данные

### Sample Input 1:

5

### Sample Output 1:

101

### Sample Input 2:

128

### Sample Output 2:

10000000

Чтобы решить это задание откройте <https://stepik.org/lesson/284101/step/15>

## 9.2 Срезы

### Шаг 1

## Тема урока: строки

1. Срезы строк
2. Изменение символов строки
3. Решение задач

**Аннотация.** Снова строковый тип данных. Учимся делать строковые срезы, а также изменять символы в строке.

### Срезы строк

В предыдущем уроке мы научились работать с конкретными символами строки с помощью индексов `[]`. Иногда нужно бывает работать с целыми частями строки, в таком случае мы используем **срезы (slices)**. Срезы похожи на комбинацию индексации и функции `range()`.

Рассмотрим строку `s = 'abcdefghijklm'`.

Положительные индексы	0	1	2	3	4	5	6	7	8	9
Строка	a	b	c	d	e	f	g	h	i	j
Отрицательные индексы	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

The diagram shows the string 'abcdefghijklm' represented as a sequence of boxes, each containing a letter. Above the string, positive indices 0 through 9 are listed, corresponding to the letters a through j respectively. Below the string, negative indices -10 through -1 are listed, corresponding to the letters m through a respectively. The indices are aligned under their respective letters.

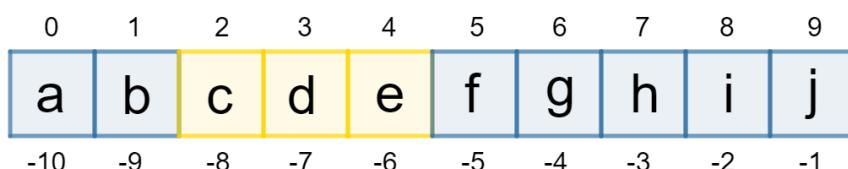
С помощью среза мы можем получить несколько символов исходной строки, создав диапазон индексов разделенных двоеточием `s[x:y]`.

Приведённый ниже код:

```
print(s[2:5])
print(s[0:6])
print(s[2:7])
```

выводит:

```
cde
abcdef
cdefg
```



0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

При построении среза `s[x:y]` первое число – это то место, где начинается срез (**включительно**), а второе – это место, где заканчивается срез (**невключительно**). Разрезая строки, мы создаем подстроку, которая по сути является строкой внутри другой строки.

### Срез до конца, от начала

Если опустить второй параметр в срезе `s[x:]` (но поставить двоеточие), то срез берется до конца строки. Аналогично если опустить первый параметр `s[:y]`, то можно взять срез от начала строки. Срез `s[:]` совпадает с самой строкой `s`.

Приведённый ниже код:

```
print(s[2:])
print(s[:7])
```

выводит:

```
cdefghij
abcdefg
```

0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1



Срез `s[:]` возвращает исходную строку.

### Отрицательные индексы в срезе

Мы также можем использовать отрицательные индексы для создания срезов. Как уже говорилось ранее, отрицательные индексы строки начинаются с `-1` и отсчитываются до достижения начала строки. При использовании отрицательных индексов **первый параметр среза должен быть меньше второго, либо должен быть пропущен**.

Приведённый ниже код:

```
print(s[-9:-4])
print(s[-3:])
print(s[:-3])
```

выводит:

```
bcde  
hij  
abcdefg
```

0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1



Удалить из строки последний символ можно при помощи среза `s[:-1]`.

### Шаг среза

Мы можем передать в срез третий необязательный параметр, который отвечает за шаг среза. К примеру, срез `s[1:7:2]` создаст строку `bdf` состоящую из каждого второго символа (индексы `1, 3, 5`, правая граница не включена в срез).

0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

### Отрицательный шаг среза

Если в качестве шага среза указать **отрицательное число**, то символы будут идти в обратном порядке.

Приведённый ниже код:

```
print(s[::-1])
```

выводит:

```
jihgfedcba
```

Приведённый ниже код:

```
print(s[1:7:2])
print(s[3::2])
print(s[:7:3])
print(s[::-2])
print(s[::-1])
print(s[::-2])
```

выводит:

```
bdf  
dfhj  
adg  
acegi  
jihgfedcba  
jhfdb
```

## Подводя итог

```
s = 'abcdefghijklm'
```

Программный код	Результат	Пояснение
s[2:5]	cde	строка состоящая из символов с индексами 2, 3, 4
s[:5]	abcde	первые пять символов строки
s[5:]	fghij	строка состоящая из символов с индексами от 5 до конца
s[-2:]	ij	последние два символа строки
s[:]	abcdefghijklm	вся строка целиком
s[1:7:2]	bdf	строка, состоящая из каждого второго символа с индексами от 1 до 6
s[::-1]	jihgfedcba	строка в обратном порядке, так как шаг отрицательный

## Изменение символа строки по индексу

Предположим, у нас есть строка `s = 'abcdefghijklm'` и мы хотим заменить символ с индексом 4 на `'X'`. Можно попытаться написать код:

```
s[4] = 'X'
```

Однако такой код не работает. В Python строки являются **неизменяемыми**, то есть мы не можем менять их содержимое с помощью индексатора.

Если мы хотим поменять какой-либо символ строки `s`, мы должны создать новую строку. Следующий код использует срезы и решает поставленную задачу:

```
s = s[:4] + 'X' + s[5:]
```

Мы создаем два среза: от начала строки до 4 символа и с 5 символа по конец строки, а между ними вставляем нужный нам символ, который встанет на 4 позицию.

## Примечания

**Примечание 1.** Синтаксис срезов строк очень похож на синтаксис функции `range()`.

**Примечание 2.** Если первый параметр среза больше второго, то срез создает пустую строку.

❤️ Happy Pythoning! 🧑

## Шаг 2

Что покажет приведённый ниже код?

```
s = 'abcdefghijklm'  
print(s[2:5])
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/2>**

## Шаг 3

Что покажет приведённый ниже код?

```
s = 'abcdefg'  
print(s[3:])
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/3>**

## Шаг 4

Что покажет приведённый ниже код?

```
s = 'abcdefg'  
print(s[:3])
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/4>**

## Шаг 5

Что покажет приведённый ниже код?

```
s = 'abcdefg'  
print(s[:])
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/5>**

## Шаг 6

Что покажет приведённый ниже код?

```
s = 'abcdefg'  
print(s[::-3])
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/6>**

## Шаг 7

Используя срезы, дополните приведённый ниже код так, чтобы он вывел первые 12 символов строки `s`.

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/7>**

## Шаг 8

Используя срезы, дополните приведённый ниже код так, чтобы он вывел последние 9 символов строки `s`.

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/8>**

## Шаг 9

Используя срезы, дополните приведённый ниже код так, чтобы он вывел каждый 7 символ строки `s` (начиная с 0-го индекса).

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/9>**

## Шаг 10

Используя срезы, дополните приведённый ниже код так, чтобы он вывел строку `s` в обратном порядке.

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/10>**

## Шаг 11

### Палиндром

На вход программе подаётся одно слово, записанное в нижнем регистре. Напишите программу, которая определяет, является ли оно палиндромом.

#### Формат входных данных

На вход программе подаётся одно слово в нижнем регистре.

#### Формат выходных данных

Программа должна вывести «YES» (без кавычек), если слово является палиндромом, или «NO» (без кавычек) в противном случае.

**Примечание.** Палиндром считается слово, которое читается одинаково в обоих направлениях. Например, слово «потоп» является палиндромом.

▶ Тестовые данные

**Sample Input 1:**

потоп

**Sample Output 1:**

YES

**Sample Input 2:**

анекдот

**Sample Output 2:**

NO

**Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/11>**

## Шаг 12

### Делаем срезы 1

На вход программе подаётся одна строка. Напишите программу, которая выводит:

1. общее количество символов в строке;
2. исходную строку, повторённую 3 раза;
3. первый символ строки;
4. первые три символа строки;
5. последние три символа строки;
6. строку в обратном порядке;
7. строку с удалённым первым и последним символами.

**Формат входных данных**

На вход программе подаётся одна строка, длина которой больше 3 символов.

**Формат выходных данных**

Программа должна вывести данные в соответствии с условием. Каждое значение выводится на отдельной строке.

► Тестовые данные 

**Sample Input 1:**

abcdefghijklmnopqrstuvwxyz

**Sample Output 1:**

```
26
abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
a
abc
xyz
zyxwvutsrqponmlkjihgfedcba
bcdefghijklmnopqrstuvwxyz
```

**Sample Input 2:**

Success is the ability to go from failure to failure without losing your enthusiasm

**Sample Output 2:**

83

Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/12>

## Шаг 13

## Делаем срезы 2

На вход программе подаётся одна строка. Напишите программу, которая выводит:

1. третий символ этой строки;
  2. предпоследний символ этой строки;
  3. первые пять символов этой строки;
  4. всю строку, кроме последних двух символов;
  5. все символы с чётными индексами;
  6. все символы с нечётными индексами;
  7. все символы в обратном порядке;
  8. все символы строки через один в обратном порядке, начиная с последнего.

## Формат входных данных

На вход программе подаётся одна строка, длина которой больше 5 символов.

## **Формат выходных данных**

Программа должна вывести данные в соответствии с условием. Каждое значение выводится на отдельной строке.

## ► Тестовые данные

## Sample Input:

abcdefghijklmnopqrstuvwxyz

## Sample Output:

```
c  
y  
abcde  
abcdefghijklmnopqrstuvwxyz  
acegikmoqsuwy  
bdfhjlnprtvxz  
zyxwvutsrqponmlkjihgfedc  
zxytrpnljhfdb
```

Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/13>

## Две половинки

На вход программе подаётся строка текста. Напишите программу, которая разрежет её на две равные части, переставит их местами и выведет на экран.

### Формат входных данных

На вход программе подаётся строка текста.

### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание.** Если длина строки нечётная, то длина первой части должна быть на один символ больше.

► Тестовые данные ●

**Sample Input 1:**

abcdef

**Sample Output 1:**

defabc

**Sample Input 2:**

abcdefg

**Sample Output 2:**

efgabcd

**Sample Input 3:**

a

**Sample Output 3:**

a

Чтобы решить это задание откройте <https://stepik.org/lesson/302627/step/14>

# 9.3 Методы строк. Часть 1

## Шаг 1

### Тема урока: строки

1. Метод `capitalize()`
2. Метод `swapcase()`
3. Метод `title()`
4. Метод `lower()`
5. Метод `upper()`
6. Решение задач

**Аннотация.** Строковый тип данных, основные методы конвертации регистра.

### Методы и функции

Мы уже знакомы с некоторыми встроенными функциями: `min()`, `max()`, `len()`, `int()`, `float()` и так далее. Метод – специализированная функция, тесно связанная с объектом. Как и функция, метод вызывается для выполнения отдельной задачи, но он вызывается для определенного объекта и “знает” о своем целевом объекте во время выполнения.

Таким образом: метод – функция, применяемая к объекту. В данном случае к строке. Метод вызывается в виде `имя_объекта.имя_метода(параметры)`.

Например, `s.find('e')` – это применение к строке `s` метода `find` с одним параметром `'e'`.

Методы строкового типа данных можно разделить на три группы:

1. Конвертация регистра;
2. Поиск и замена;
3. Классификация символов.

### Конвертация регистра

Методы в этой группе выполняют преобразование регистра для строк.

#### Метод `capitalize()`

Метод `capitalize()` возвращает копию строки `s`, в которой первый символ имеет верхний регистр, а все остальные символы имеют нижний регистр.

Приведённый ниже код:

```
s = 'fo0 BaR BAZ quX'  
print(s.capitalize())
```

выводит:

```
Foo bar baz qux
```

Символы, не являющиеся буквами алфавита, остаются неизменными.

Приведённый ниже код:

```
s = 'foo123#BAR#.'  
print(s.capitalize())
```

выводит:

```
Foo123#bar#.
```

### Метод swapcase()

Метод `swapcase()` возвращает копию строки `s`, в которой все символы, имеющие верхний регистр, преобразуются в символы нижнего регистра и наоборот.

Приведённый ниже код:

```
s = 'FOO Bar 123 baz qUX'  
print(s.swapcase())
```

выводит:

```
foo bAR 123 BAZ Qux
```

### Метод title()

Метод `title()` возвращает копию строки `s`, в которой первый символ каждого слова переводится в верхний регистр.

Приведённый ниже код:

```
s = 'the sun also rises'  
print(s.title())
```

выводит:

```
The Sun Also Rises
```

Этот метод использует довольно простой алгоритм: он не пытается различить важные и неважные слова и не обрабатывает аббревиатуры и апострофы.

Приведённый ниже код:

```
s = "what's happened to ted's IBM stock?"  
print(s.title())
```

выводит:

```
What'S Happened To Ted'S Ibm Stock?
```

### Метод lower()

Метод `lower()` возвращает копию строки `s`, в которой все символы имеют нижний регистр.

Приведённый ниже код:

```
s = 'FOO Bar 123 baz qUX'  
print(s.lower())
```

выводит:

```
foo bar 123 baz qux
```

## Метод upper()

Метод `upper()` возвращает копию строки `s`, в которой все символы имеют верхний регистр.

Приведённый ниже код:

```
s = 'FOO Bar 123 baz qUX'  
print(s.upper())
```

выводит:

```
F00 BAR 123 BAZ QUX
```



Одно очень важное замечание о методах данной категории состоит в том, что они не изменяют исходную строку.

Если вы хотите изменить строку `s`, нужно написать код: `s = s.lower()`. На самом деле тут вы создаёте совсем другой объект в памяти компьютера, просто он со старым названием `s`.

## Примечание

Англо-русский словарик:

`capitalize` – писать прописными буквами, закрепить.

`swapcase` – обменять регистр. `swap` – гл. обмениваться, `case` – случай, регистр, падеж, дело, расследование...

`title` – заголовок, титул.

`lower` – нижний.

`upper` – верхний.

❤️ Happy Pythoning! 🐍

## Шаг 2

Что покажет приведённый ниже код?

```
s = 'i Learn Python language'  
print(s.capitalize())
```

Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/2>

## Шаг 3

Что покажет приведённый ниже код?

```
s = 'i LEARN Python LAnguaGE'  
print(s.lower())
```

Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/3>

## Шаг 4

Что покажет приведённый ниже код?

```
s = '$12344%#$@!'
print(s.lower())
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/4>**

## Шаг 5

Что покажет приведённый ниже код?

```
s1 = 'a'
s2 = s1.upper()
print(s1, s2)
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/5>**

## Шаг 6

Что покажет приведённый ниже код?

```
s = 'i LEARN Python LAnguaGE'
print(s.upper())
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/6>**

## Шаг 7

Что покажет приведённый ниже код?

```
s = 'i LEARN Python LAnguaGE'
print(s.swapcase())
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/7>**

## Шаг 8

### Заглавные буквы

На вход программе подаётся строка, состоящая из имени и фамилии человека, разделённых одним пробелом. Напишите программу, которая проверяет, что имя и фамилия начинаются с заглавной буквы.

## Формат входных данных

На вход программе подаётся строка.

## Формат выходных данных

Программа должна вывести «YES» (без кавычек), если имя и фамилия начинаются с заглавной буквы, или «NO» (без кавычек) в противном случае.

**Примечание.** Стока содержит только буквы и символ пробела.

► Тестовые данные 

**Sample Input 1:**

chris alan

**Sample Output 1:**

NO

**Sample Input 2:**

Chris Alan

**Sample Output 2:**

YES

**Sample Input 3:**

chris Alan

**Sample Output 3:**

NO

Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/8>

Шаг 9

## sWAP cASE

На вход программе подаётся строка. Напишите программу, которая меняет регистр символов – заменяет все строчные символы заглавными и наоборот.

## Формат входных данных

На вход программе подаётся строка.

## Формат выходных данных

Программа должна вывести строку в соответствии с условием задачи.

► Тестовые данные 

**Sample Input 1:**

HTTPS://pygen.RU/

**Sample Output 1:**

https://PYGEN.ru/

**Sample Input 2:**

Www.sTepik.com

**Sample Output 2:**

**Sample Input 3:**

Pythonist 2

**Sample Output 3:**

pYTHONIST 2

Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/9>

## Шаг 10

### Хороший оттенок

На вход программе подаётся строка текста. Напишите программу, которая определяет, является ли оттенок текста хорошим или нет. Текст имеет хороший оттенок, если содержит подстроку «хорош» (без кавычек) во всевозможных регистрах.

**Формат входных данных**

На вход программе подаётся строка текста.

**Формат выходных данных**

Программа должна вывести «YES» (без кавычек), если текст имеет хороший оттенок, или «NO» (без кавычек) в противном случае.

**Примечание.** Текст, содержащий «хорош», «ХОРОШ», «Хорош», «хОРОШ» и так далее также имеет хороший оттенок.

► Тестовые данные 

**Sample Input 1:**

я очень хороший текст =)

**Sample Output 1:**

YES

**Sample Input 2:**

оыралоывало ХОРОШвмсва выарлвօ3կգ834

**Sample Output 2:**

YES

**Sample Input 3:**

Цвет настроения синий

**Sample Output 3:**

NO

Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/10>

### Нижний регистр

На вход программе подаётся строка. Напишите программу, которая подсчитывает количество буквенных символов в нижнем регистре.

#### Формат входных данных

На вход программе подаётся строка.

#### Формат выходных данных

Программа должна вывести количество буквенных символов в нижнем регистре.

► Тестовые данные 

#### Sample Input 1:

abcABCD12345

#### Sample Output 1:

3

#### Sample Input 2:

gggggggg1212321ABDCEFCE

#### Sample Output 2:

8

#### Sample Input 3:

2376423745dhdhdPPPP

#### Sample Output 3:

5

Чтобы решить это задание откройте <https://stepik.org/lesson/296416/step/11>

## 9.4 Методы строк. Часть 2

### Шаг 1

## Тема урока: строки

1. Методы `count()`
2. Метод `startswith()`
3. Метод `endswith()`
4. Метод `find()`
5. Метод `rfind()`
6. Метод `index()`
7. Метод `rindex()`
8. Метод `strip()`
9. Метод `lstrip()`
10. Метод `rstrip()`
11. Метод `replace()`

**Аннотация.** Строковый тип данных, основные методы поиска и замены.

### Поиск и замена

Методы поиска и замены строк внутри других строк.

Каждый метод в этой группе поддерживает необязательные аргументы `<start>` и `<end>`. Как и в строковых срезах, действие метода ограничено частью исходной строки, начинающейся с позиции символа `<start>` и продолжающейся вплоть до позиции символа `<end>`, но не включающей её. Если параметр `<start>` указан, а параметр `<end>` нет, то метод применяется к части исходной строки от `<start>` до конца строки. Если параметры не заданы, то подразумевается, что `<start> = 0`, `<end> = len(s)`.

#### Метод `count()`

Метод `count(<sub>, <start>, <end>)` считает количество **непересекающихся** вхождений подстроки `<sub>` в исходную строку `s`.

Приведённый ниже код:

```
s = 'foo goo moo'
print(s.count('oo'))
print(s.count('oo', 0, 8)) # подсчет с 0 по 7 символов
```

выводит:

```
3
2
```

#### Метод `startswith()`

Метод `startswith(<suffix>, <start>, <end>)` определяет, **начинается** ли исходная строка `s` подстрокой `<suffix>`. Метод возвращает значение `True`, если исходная строка начинается с подстроки `<suffix>`, или `False` в противном случае.

Приведённый ниже код:

```
s = 'foobar'  
print(s.startswith('foo'))  
print(s.startswith('baz'))
```

выводит:

```
True  
False
```

## Метод endswith()

Метод `endswith(<suffix>, <start>, <end>)` определяет, **оканчивается ли** исходная строка `s` подстрокой `<suffix>`. Метод возвращает значение `True`, если исходная строка оканчивается на подстроку `<suffix>`, или `False` в противном случае.

Приведённый ниже код:

```
s = 'foobar'  
print(s.endswith('bar'))  
print(s.endswith('baz'))
```

выводит:

```
True  
False
```

## Методы find(), rfind()

Метод `find(<sub>, <start>, <end>)` находит **индекс первого вхождения** подстроки `<sub>` в исходной строке `s`. Если строка `s` не содержит подстроки `<sub>`, то метод возвращает значение `-1`. Мы можем использовать данный метод наравне с оператором `in` для проверки: содержит ли заданная строка некоторую подстроку или нет.

Приведённый ниже код:

```
s = 'foo bar foo baz foo qux'  
print(s.find('foo'))  
print(s.find('bar'))  
print(s.find('qu'))  
print(s.find('python'))
```

выводит:

```
0  
4  
20  
-1
```

Метод `rfind(<sub>, <start>, <end>)` идентичен методу `find(<sub>, <start>, <end>)`, за тем исключением, что он ищет первое вхождение подстроки `<sub>`, начиная с конца строки `s`.

## Методы index(), rindex()

Метод `index(<sub>, <start>, <end>)` идентичен методу `find(<sub>, <start>, <end>)`, за тем исключением, что он **вызывает ошибку** `ValueError: substring not found` во время выполнения программы, если подстрока `<sub>` не найдена.

Метод `rindex(<sub>, <start>, <end>)` идентичен методу `index(<sub>, <start>, <end>)`, за тем исключением, что он ищет первое вхождение подстроки `<sub>`, начиная с конца строки `s`.



Методы `find()` и `rfind()` являются более безопасными, чем `index()` и `rindex()`, так как не приводят к возникновению ошибки во время выполнения программы.

## Метод strip()

Метод `strip()` возвращает копию строки `s`, у которой удалены все пробелы, стоящие **в начале и конце** строки.

Приведённый ниже код:

```
s = '    foo bar foo baz foo qux    '
print(s.strip())
```

выводит:

```
foo bar foo baz foo qux
```

## Метод lstrip()

Метод `lstrip()` возвращает копию строки `s`, у которой удалены все пробелы, стоящие **в начале** строки.

Приведённый ниже код:

```
s = '    foo bar foo baz foo qux    '
print(s.lstrip())
```

выводит (символом `_` обозначены пробелы):

```
foo bar foo baz foo qux_____
```

## Метод rstrip()

Метод `rstrip()` возвращает копию строки `s`, у которой удалены все пробелы, стоящие **в конце** строки.

Приведённый ниже код:

```
s = '    foo bar foo baz foo qux    '
print(s.rstrip())
```

выводит (символом `_` обозначены пробелы):

```
_____foo bar foo baz foo qux
```

## Метод replace()

Метод `replace(<old>, <new>)` возвращает копию `s` **со всеми** вхождениями подстроки `<old>`, заменёнными на `<new>`.

Приведённый ниже код:

```
s = 'foo bar foo baz foo qux'
print(s.replace('foo', 'grault'))
```

выводит:

```
grault bar grault baz grault qux
```

Метод `replace()` может принимать необязательный третий аргумент `<count>`, который определяет количество замен.

Приведённый ниже код:

```
s = 'foo bar foo baz foo qux'
print(s.replace('foo', 'grault', 2))
```

выводит:

```
grault bar grault baz foo qux
```

## Примечания

**Примечание.** Методы `strip()`, `lstrip()`, `rstrip()` могут принимать на вход необязательный аргумент `<chars>`. Необязательный аргумент `<chars>` – это строка, которая определяет набор символов для удаления.

Приведённый ниже код:

```
s = '-++-+abc+-+-'  
  
print(s.strip('+-'))  
print(s.rstrip('+-'))  
print(s.lstrip('+-'))
```

выводит:

```
abc  
-++-+abc  
abc+-+-
```

❤️ Happy Pythoning! 🧑

## Шаг 2

Что покажет приведённый ниже код?

```
s = 'aabbaAccDDaa'  
s = s.lower()  
print(s.count('a'))
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/2>

## Шаг 3

Что покажет приведённый ниже код?

```
s = 'www.stepik.org'  
print(s.startswith('www'))
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/3>

## Шаг 4

Что покажет приведённый ниже код?

```
s = 'www.stepik.org'  
print(s.endswith('.ru'))
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/4>

## Шаг 5

Что покажет приведённый ниже код?

```
s = 'I learn Python language. Python - awesome!'
print(s.find('Python'))
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/5>

## Шаг 6

Что покажет приведённый ниже код?

```
s = '      I learn Python language
print(s.strip())
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/6>

## Шаг 7

Что покажет приведённый ниже код?

```
s = 'abcdababa'
print(s.replace('ab', '123'))
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/7>

## Шаг 8

### Количество слов

На вход программе подаётся строка текста, состоящая из слов, разделённых ровно одним пробелом. Напишите программу, которая подсчитывает количество слов в ней.

#### Формат входных данных

На вход программе подаётся строка текста.

#### Формат выходных данных

Программа должна вывести количество слов.

**Примечание.** Стока текста не содержит пробелов в начале и конце.

► Тестовые данные

**Sample Input 1:**

Hello world

**Sample Output 1:**

2

**Sample Input 2:**

Python

**Sample Output 2:**

1

**Sample Input 3:**

In 2010, someone paid 10k Bitcoin for two pizzas.

**Sample Output 3:**

9

**Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/8>**

**Шаг 9**

## Минутка генетики

На вход программе подаётся строка генетического кода, состоящая из букв А (аденин), Г (гуанин), Ц (цитозин) и Т (тимин). Напишите программу, которая подсчитывает сколько аденина, гуанина, цитозина и тимина входит в данную строку генетического кода.

### Формат входных данных

На вход программе подаётся строка генетического кода, состоящая из символов А, Г, Ц, Т, а, г, ц, т.

### Формат выходных данных

Программа должна вывести, сколько гуанина, тимина, цитозина, аденина входит в данную строку генетического кода.

**Примечание.** Стока не содержит других символов, кроме А, Г, Ц, Т, а, г, ц, т.

► Тестовые данные

**Sample Input 1:**

АаагГГЦЦцТТtttT

**Sample Output 1:**

Аденин: 3  
Гуанин: 2  
Цитозин: 3  
Тимин: 5

**Sample Input 2:**

ааггццттААГГЦЦтт

**Sample Output 2:**

Аденин: 4  
Гуанин: 4  
Цитозин: 4  
Тимин: 4

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/9>

## Шаг 10

### Очень странные дела

Джим Хоппер с помощью радиоприёмника пытается получить сообщение Оди. На приёмник ему поступает  $n$  различных последовательностей кода Морзе. Декодировав их, он получает последовательности из цифр и букв строчного латинского алфавита. При этом **только в сообщениях Оди** содержится число 11, причём минимум 3 раза. Помогите определить Джиму количество сообщений от Оди.

#### Формат входных данных

В первой строке подаётся число  $n$  – количество сообщений, в последующих  $n$  строках вводятся сами сообщения.

#### Формат выходных данных

Программа должна вывести количество сообщений от Оди.

**Примечание.** Обратите внимание, что в сообщениях Оди вхождения числа 11 должны быть **непересекающимися**. Другими словами, если мы нашли вхождение числа 11, то следующее вхождение должно начинаться строго после окончания предыдущего. Например, в строке '111' содержится одна такая последовательность, в то время как в '1111' их уже две.

► Тестовые данные 

#### Sample Input 1:

```
3
11helpme11jim11
avengers141414atta11ck
k1lg0re11111l
```

#### Sample Output 1:

```
1
```

#### Sample Input 2:

```
5
eowi11fjoigei23do11i23gf2983fj19f1120f92f23
i2oj2fjoerfpwfmkewfewkekef
wewewewew111wweeweweewew11w
111111
krewjrgeorgjogergiurg
```

#### Sample Output 2:

```
2
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/10>

## Шаг 11

### Количество цифр

На вход программе подаётся строка текста. Напишите программу, которая подсчитывает количество цифр в данной строке.

#### Формат входных данных

На вход программе подаётся строка текста.

#### Формат выходных данных

Программа должна вывести количество цифр в данной строке.

► Тестовые данные 

**Sample Input 1:**

nezabud dl-6

**Sample Output 1:**

1

**Sample Input 2:**

l33t 3301

**Sample Output 2:**

6

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/11>

## Шаг 12

.com or .ru 

На вход программе подаётся строка текста. Напишите программу, которая проверяет, что строка заканчивается подстрокой

.com или .ru.

#### Формат входных данных

На вход программе подаётся строка текста.

#### Формат выходных данных

Программа должна вывести «YES» (без кавычек), если введённая строка заканчивается подстрокой .com или .ru, или «NO» (без кавычек) в противном случае.

► Тестовые данные 

**Sample Input 1:**

www.stepik.org

**Sample Output 1:**

NO

**Sample Input 2:**

www.google.com

**Sample Output 2:**

YES

**Sample Input 3:**

www.yandex.ru

**Sample Output 3:**

YES

**Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/12>**

**Шаг 13**

## **Самый частотный символ**

На вход программе подаётся строка текста. Напишите программу, которая выводит на экран символ, который появляется наиболее часто.

### **Формат входных данных**

На вход программе подаётся строка текста. Текст может содержать строчные и заглавные буквы английского и русского алфавита, а также цифры.

### **Формат выходных данных**

Программа должна вывести символ, который появляется наиболее часто.

**Примечание 1.** Если таких символов несколько, следует вывести последний по порядку символ.

**Примечание 2.** Следует различать заглавные и строчные буквы, а также буквы русского и английского алфавита.

► Тестовые данные ●

**Sample Input 1:**

aaaabbc

**Sample Output 1:**

a

**Sample Input 2:**

abaabbcccc

**Sample Output 2:**

c

**Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/13>**

**Шаг 14**

## **Первое и последнее вхождение**

На вход программе подаётся строка текста. Если в этой строке буква «f» встречается только один раз, выведите её индекс. Если она встречается два и более раза, выведите индексы её первого и последнего вхождения на одной строке, разделённые символом пробела. Если буква «f» в данной строке не встречается, следует вывести «NO» (без кавычек).

## Формат входных данных

На вход программе подаётся строка текста.

## Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

► Тестовые данные 

**Sample Input 1:**

abcdefg

**Sample Output 1:**

5

**Sample Input 2:**

abcdefgfhfabc

**Sample Output 2:**

5 9

**Sample Input 3:**

abcd

**Sample Output 3:**

NO

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/14>

Шаг 15

## Удаление фрагмента

На вход программе подаётся строка текста, в которой буква «h» встречается минимум два раза. Напишите программу, которая удаляет из этой строки первое и последнее вхождение буквы «h», а также все символы, находящиеся между ними.

## Формат входных данных

На вход программе подаётся строка текста.

## Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

► Тестовые данные 

**Sample Input 1:**

ahahahahaha

**Sample Output 1:**

aa

**Sample Input 2:**

hh

**Sample Output 2:**

Чтобы решить это задание откройте <https://stepik.org/lesson/303083/step/15>

# 9.5 Методы строк. Часть 3

## Шаг 1

### Тема урока: строки

1. Методы строк `isalnum()`
2. Метод `isalpha()`
3. Метод `isdigit()`
4. Метод `islower()`
5. Метод `isupper()`
6. Метод `isspace()`
7. Решение задач

**Аннотация.** Строковый тип данных, основные методы классификации символов.

### Классификация символов

Методы в этой группе классифицируют строку на основе содержащихся в ней символов.

#### Метод `isalnum()`

Метод `isalnum()` определяет, состоит ли исходная строка из буквенно-цифровых символов. Метод возвращает значение `True`, если исходная строка является непустой и состоит **только** из буквенно-цифровых символов, или `False` в противном случае.

Приведённый ниже код:

```
s1 = 'abc123'  
s2 = 'abc$*123'  
s3 = ''  
  
print(s1.isalnum())  
print(s2.isalnum())  
print(s3.isalnum())
```

выводит:

```
True  
False  
False
```

Обратите внимание, что метод `isalnum()` возвращает значение `True` даже в том случае, когда строка состоит только из буквенных или только из цифровых символов.

Приведённый ниже код:

```
s1 = 'BEEGEEK'  
s2 = '2202'  
  
print(s1.isalnum())  
print(s2.isalnum())
```

выводит:

```
True  
True
```

## Метод `isalpha()`

Метод `isalpha()` определяет, состоит ли исходная строка из буквенных символов. Метод возвращает значение `True`, если исходная строка является непустой и состоит **только** из буквенных символов, или `False` в противном случае.

Приведённый ниже код:

```
s1 = 'ABCabc'  
s2 = 'abc123'  
s3 = ''  
  
print(s1.isalpha())  
print(s2.isalpha())  
print(s3.isalpha())
```

выводит:

```
True  
False  
False
```

## Метод `isdigit()`

Метод `isdigit()` определяет, состоит ли исходная строка **только** из цифровых символов. Метод возвращает значение `True`, если исходная строка является непустой и состоит **только** из цифровых символов, или `False` в противном случае.

Приведённый ниже код:

```
s1 = '1234567'  
s2 = 'abc123'  
s3 = ''  
  
print(s1.isdigit())  
print(s2.isdigit())  
print(s3.isdigit())
```

выводит:

```
True  
False  
False
```

## Метод `islower()`

Метод `islower()` определяет, являются ли **все** буквенные символы исходной строки строчными (имеют нижний регистр). Метод возвращает значение `True`, если все буквенные символы исходной строки являются строчными, или `False` в противном случае.

Приведённый ниже код:

```
s1 = 'abc'  
s2 = 'abc1$d'  
s3 = 'Abc1$D'  
  
print(s1.islower())  
print(s2.islower())  
print(s3.islower())
```

выводит:

```
True  
True  
False
```

Обратите внимание, что метод `islower()` **игнорирует все небуквенные символы**.

Приведённый ниже код:

```
print('1234'.islower())
print('+-*/'.islower())
print('ab#%'.islower())
```

выводит:

```
False
False
True
```

В первом и втором случаях у нас в строках отсутствуют буквенные символы в нижнем регистре, поэтому метод `islower()` и возвращает значение `False`.

### Метод `isupper()`

Метод `isupper()` определяет, являются ли **все** буквенные символы исходной строки заглавными (имеют верхний регистр).

Метод возвращает значение `True`, если все буквенные символы исходной строки являются заглавными, или `False` в противном случае.

Приведённый ниже код:

```
s1 = 'ABC'
s2 = 'ABC1$D'
s3 = 'Abc1$D'

print(s1.isupper())
print(s2.isupper())
print(s3.isupper())
```

выводит:

```
True
True
False
```

Обратите внимание, что метод `isupper()` **игнорирует все небуквенные символы**.

Приведённый ниже код:

```
print('5678'.isupper())
print('!?_&'.isupper())
print('AB%$'.isupper())
```

выводит:

```
False
False
True
```

В первом и втором случаях у нас в строках отсутствуют буквенные символы в верхнем регистре, поэтому метод `isupper()` и возвращает значение `False`.

### Метод `isspace()`

Метод `isspace()` определяет, состоит ли исходная строка **только** из пробельных символов. Метод возвращает значение `True`, если строка состоит только из пробельных символов, или `False` в противном случае.

Приведённый ниже код:

```
s1 = ''  
s2 = 'abc1$d'  
  
print(s1isspace())  
print(s2isspace())
```

выводит:

```
True  
False
```

Для пустой строки метод `isspace()` также возвращает `False`, так как в этом случае строка не состоит из пробельных символов.

Приведённый ниже код:

```
s1 = ''  
print(s1isspace())
```

выводит:

```
False
```



## Шаг 2

Что покажет приведённый ниже код?

```
s = 'aabbAA111ccDDaa'  
print(s.isalnum())  
print(s.isalpha())  
print(s.isdigit())
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/2>

## Шаг 3

Что покажет приведённый ниже код?

```
print('Cyberpunk 2077'.isalnum())
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/3>

## Шаг 4

Что покажет приведённый ниже код?

```
print('Cyberpunk'.isalnum())  
print('2077'.isalnum())
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/4>**

## Шаг 5

Что покажет приведённый ниже код?

```
s = 'aabb!@#$11cc'  
print(s.islower())
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/5>**

## Шаг 6

Что покажет приведённый ниже код?

```
s = 'AAb!@#$11CC'  
print(s.isupper())
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/6>**

## Шаг 7

Что покажет приведённый ниже код?

```
print('2024-05-19'.islower())  
print('2024-05-19'.isupper())
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/7>**

## Шаг 8

Что покажет приведённый ниже код?

```
s = '      abbc      '  
print(s.isspace())
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/8>**

### Плохие комментарии

На платформе Stepik пользователи оставляют комментарии, но не все из них соответствуют правилам. Так, например, модератор Сэм считает неуместными комментариями те, которые представляют собой пустую строку или состоят только из пробелов. Подобные комментарии он удаляет – нечего засорять курс бесполезным материалом!

Ваша задача – написать программу, которая поможет Сэму проверять комментарии. Программа должна принимать на вход натуральное число  $n$ , а затем  $n$  строк, представляющих тексты комментариев. Для каждого комментария ваша программа должна выводить номер этого комментария (начиная с 1), затем двоеточие ( : ), затем через пробел его текст или сообщение «COMMENT SHOULD BE DELETED» (без кавычек), если комментарий должен быть удалён Сэмом.

#### Формат входных данных

На вход программе подаются натуральное число  $n$ , а затем –  $n$  строк.

#### Формат выходных данных

Для каждого комментария программа должна вывести его текст или сообщение «COMMENT SHOULD BE DELETED» (без кавычек), если комментарий должен быть удалён.

#### ► Тестовые данные

##### Sample Input 1:

```
4
Не проходит 7 тест, что у меня нет так?
Авторы, вы надоели уже со своей математикой и шахматами!

Почему у меня выходит ошибка IndexError?
```

##### Sample Output 1:

```
1: Не проходит 7 тест, что у меня нет так?
2: Авторы, вы надоели уже со своей математикой и шахматами!
3: COMMENT SHOULD BE DELETED
4: Почему у меня выходит ошибка IndexError?
```

##### Sample Input 2:

```
4

Почему мне не начислили баллы за задачу?! Я ее с первого раза решила
Я крутой, я решил контрольную за 7 минут..
```

##### Sample Output 2:

```
1: COMMENT SHOULD BE DELETED
2: COMMENT SHOULD BE DELETED
3: Почему мне не начислили баллы за задачу?! Я ее с первого раза решила
4: Я крутой, я решил контрольную за 7 минут..
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/9>

## Шаг 10

### Автомобильный номер 🚗 🇷🇺

В службе по дорожному движению решили оптимизировать процесс создания автомобильных номеров: вместо человека генерацию автомобильных номеров поручили некоторой GPT (модели машинного обучения). Как мы знаем, искусственный интеллект ещё сыротав и делает много ошибок, поэтому его результаты следует тщательно проверять. Корректный автомобильный номер (в России) имеет следующий формат:



Напишите программу, которая принимает на вход строку и проверяет, является ли эта строка корректным автомобильным номером. Программа должна вывести «YES» (без кавычек), если искусственный интеллект справился со своей задачей, или «NO» (без кавычек) в противном случае. В нашей задаче корректным автомобильным номером будем считать следующие форматы:

```
<БУКВА><ЦИФРА><ЦИФРА><ЦИФРА><БУКВА><БУКВА>_<ЦИФРА><ЦИФРА>
```

```
<БУКВА><ЦИФРА><ЦИФРА><ЦИФРА><БУКВА><БУКВА>_<ЦИФРА><ЦИФРА><ЦИФРА>
```

где <ЦИФРА> – это любая цифра, а <БУКВА> – это одна из букв кириллицы АВЕКМНОРСТУХ .

#### Формат входных данных

На вход программе подаётся одна строка – сгенерированный ИИ автомобильный номер.

#### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание.** Подробнее почитать про автомобильные номера можно по ссылке

(

#### ► Тестовые данные

##### Sample Input 1:

```
A123BC_45
```

##### Sample Output 1:

```
YES
```

##### Sample Input 2:

```
АЯ123В_45
```

##### Sample Output 2:

```
NO
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/10>

## Шаг 11

### Проверь никнейм

Во время собеседования вам предложили решить задачу на валидацию имени пользователя. Пользователь пытается создать никнейм для своего аккаунта в соцсети Y. Правила для корректного никнейма в соцсети Y следующие:

- никнейм должен начинаться с символа `@`
- никнейм должен содержать от 5 до 15 (включительно) символов (включая первый символ `@`)
- никнейм должен содержать только **строчные** буквы и цифры (помимо первого символа `@`)

Напишите программу, которая выводит «Correct» (без кавычек), если никнейм соответствует **всем** вышеприведенным правилам, или «Incorrect» (без кавычек) в противном случае.

#### Формат входных данных

На вход программе подаётся одна строка – никнейм.

#### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание.** Обратите внимание, что никнейму необязательно содержать строчные буквы и цифры **одновременно**, никнейм может содержать только строчные буквы или только цифры (помимо первого символа `@`). Например, следующие никнеймы считаются корректными:

```
@duncan  
@1111
```

#### ► Тестовые данные

##### Sample Input 1:

```
@paul_atreides
```

##### Sample Output 1:

```
Incorrect
```

##### Sample Input 2:

```
@chani7
```

##### Sample Output 2:

```
Correct
```

Чтобы решить это задание откройте <https://stepik.org/lesson/303084/step/11>

## 9.6 Форматирование строк

### Шаг 1

## Тема урока: форматирование строк

1. Строковый метод `format()`

2. f-строки

**Аннотация.** Строковый тип данных, основные методы форматирования строк.

### Метод `format()`

Хранить строки в переменных удобно, но часто бывает необходимо собирать строки из других объектов (строк, чисел и т.д.) и выполнять с ними нужные манипуляции. Для этой цели можно воспользоваться механизмом **форматирования строк**.

Рассмотрим следующий код:

```
birth_year = 1992
text = 'My name is Timur, I was born in ' + birth_year
print(text)
```

Такой код приводит к ошибке во время выполнения программы, поскольку мы пытаемся сложить число и строку:

```
TypeError: can only concatenate str (not "int") to str
```

Для решения такой проблемы мы можем использовать функцию `str()`, которая преобразует числовое значение в строку.

Приведенный ниже код:

```
birth_year = 1992
text = 'My name is Timur, I was born in ' + str(birth_year)
print(text)
```

выводит:

```
My name is Timur, I was born in 1992
```

Такой код работает, но каждый раз преобразовывать число в строку не очень удобно. Для более наглядного форматирования мы можем использовать строковый метод `format()`.

Предыдущий код можно переписать в виде:

```
birth_year = 1992
text = 'My name is Timur, I was born in {}'.format(birth_year)
print(text)
```

Мы передаем необходимые параметры методу `format()`, а Python ставит их вместо фигурных скобок `{}` – **заполнителей**. Мы можем создавать сколько угодно заполнителей.

Приведенный ниже код:

```
birth_year = 1992
name = 'Timur'
profession = 'math teacher'
text = 'My name is {}, I was born in {}, I work as a {}.'.format(name, birth_year, profession)

print(text)
```

выводит:

```
My name is Timur, I was born in 1992, I work as a math teacher.
```

Для наглядности и гибкости форматирования мы можем использовать порядковый номер в заполнителе: `{0}`, `{1}`, `{2}` и т.д. Такой номер определяет позицию параметра, переданного методу `format()` (нумерация начинается с нуля):

```
birth_year = 1992
name = 'Timur'
profession = 'math teacher'
text = 'My name is {0}, I was born in {1}, I work as a {2}.format(name, birth_year, profession)

print(text)
```

Параметр `name` встает в заполнителе `{0}`, параметр `birth_year` – в заполнителе `{1}` и т.д. Мы можем использовать одно и то же число в нескольких заполнителях или не использовать совсем, а также мы можем использовать числа в разном порядке.

Приведенный ниже код:

```
name = 'Timur'
city = 'Moscow'
text1 = 'My name is {0}-{0}-{0}!'.format(name, city)
text2 = '{1} is my city and {0} is my name!'.format(name, city)

print(text1)
print(text2)
```

выводит:

```
My name is Timur-Timur-Timur!
Moscow is my city and Timur is my name!
```

## f-строки

Метод `format()` хорошо справляется с задачей форматирования строк, однако если параметров много, то код может показаться немного избыточным.

Приведенный ниже код:

```
first_name = 'Taylor'
second_name = 'Swift'
country = 'USA'
birth_date = '1989/12/13'
birth_place = 'West Reading, Pennsylvania'
text = '{0} {1} is a very famous singer from the {2}. She was born on {3} in {4}.format(first_name,
second_name, country, birth_date, birth_place)

print(text)
```

выводит:

```
Taylor Swift is a very famous singer from the USA. She was born on 1989/12/13 in West Reading,
Pennsylvania.
```

В Python 3.6 появилась новая разновидность строк – f-строки. Если поставить перед строкой префикс `f`, в заполнители можно будет включить код, например, имя переменной или любые другие выражения. f-строки обеспечивают чистый и интуитивно понятный способ форматирования строк.

Предыдущий код можно записать в виде:

```
first_name = 'Taylor'  
last_name = 'Swift'  
country = 'USA'  
birth_date = '1989/12/13'  
birth_place = 'West Reading, Pennsylvania'  
text = f'{first_name} {last_name} is a very famous singer from the {country}. She was born on {birth_date}  
in {birth_place}.'  
  
print(text)
```

На место заполнителя `{first_name}` встает значение переменной `first_name`, на место заполнителя `{last_name}` встает значение переменной `last_name` и т.д.

Как уже говорилось, помимо переменных в заполнителях f-строк мы можем использовать выражения.

Приведенный ниже код:

```
print(f'5 + 2 = {5 + 2}')  
print(f'5 - 2 = {5 - 2}')  
print(f'5 * 2 = {5 * 2}')  
print(f'5 / 2 = {5 / 2}')
```

выводит:

```
5 + 2 = 7  
5 - 2 = 3  
5 * 2 = 10  
5 / 2 = 2.5
```

## Примечания

**Примечание 1.** Почитать подробнее про форматирование строк можно в официальной документации по [ссылке](https://docs.python.org/3/library/string.html#formatstrings) (<https://docs.python.org/3/library/string.html#formatstrings>). Отдельно можно почитать на английском языке про метод `format()` и f-строки по ссылке (<https://docs.python.org/3/library/stdtypes.html#str.format>) и по [ссылке](https://docs.python.org/3/reference/lexical_analysis.html#f-strings) ([https://docs.python.org/3/reference/lexical\\_analysis.html#f-strings](https://docs.python.org/3/reference/lexical_analysis.html#f-strings)).

**Примечание 2.** На русском языке можно почитать про форматирование строк по [ссылке](https://docs-python.ru/tutorial/operatsii-tekstovymi-strokami-str-python/metod-str-format/) (<https://docs-python.ru/tutorial/operatsii-tekstovymi-strokami-str-python/metod-str-format/>) и по [ссылке](https://docs-python.ru/tutorial/operatsii-tekstovymi-strokami-str-python/stroki-formatirovannye-stroki/) (<https://docs-python.ru/tutorial/operatsii-tekstovymi-strokami-str-python/stroki-formatirovannye-stroki/>).

**Примечание 3.** Про эволюцию форматирования в Python вы можете почитать в нашем официальном блоге на Хабре по [ссылке](https://habr.com/ru/articles/828396/) (<https://habr.com/ru/articles/828396/>).

**Примечание 4.** До версии Python 3.12 повторное использование тех же кавычек, что и окружающие f-строку, вызывает ошибку синтаксиса.

Приведенный ниже код:

```
text = f'{{'London'}} is the capital of {'England'} and the {'United Kingdom'}}
```

приводит к возникновению ошибки:

```
SyntaxError: f-string: expecting '}'
```

В версии Python 3.12 это ограничение сняли. Обратите внимание, что на данный момент на Stepik недоступна версия Python 3.12, и если вы копируете код из своей IDE (в которой, возможно, уже стоит Python версии 3.12 и выше), пожалуйста, проверяйте, что вы не используете повторно одни кавычки.

❤️ Happy Pythoning! 🧑

## Шаг 2

Что покажет приведённый ниже код?

```
planet = 'Arrakis'  
bad_guys = 'Harkonnens'  
text = 'The desert planet {}, rich in valuable spice, is exploited by cruel {}'.format(planet, bad_guys)  
  
print(text)
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/2>

## Шаг 3

Что покажет приведённый ниже код?

```
name = 'Leto Atreides'  
planet = 'Caladan'  
text = 'Duke {1} is the ruler of the planet {0}'.format(planet, name)  
  
print(text)
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/3>

## Шаг 4

Что покажет приведённый ниже код?

```
name = 'Dune'  
text = f'The novel "{name}" was published in 1965 by Frank Herbert.'  
  
print(text)
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/4>

## Шаг 5

Что покажет приведённый ниже код?

```
name = 'Imperium'  
text = 'For the {name} spice is used by the navigators to find safe paths between the stars.'  
  
print(text)
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/5>**

## Шаг 6

Что покажет приведённый ниже код (в версиях Python **до** 3.12)?

```
text = f'By Imperial decree, Leto Atreides is now the fief of the planet {"Arrakis"}.'  
print(text)
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/6>**

## Шаг 7

Используя метод `format()`, дополните приведённый ниже код так, чтобы он вывел текст:

```
In 2010, someone paid 10k Bitcoin for two pizzas.
```

**Sample Input:**

**Sample Output:**

```
In 2010, someone paid 10k Bitcoin for two pizzas.
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/7>**

## Шаг 8

Используя f-строку, дополните приведённый ниже код так, чтобы он вывел текст:

```
In 2010, someone paid 10K Bitcoin for two pizzas.
```

**Sample Input:**

**Sample Output:**

```
In 2010, someone paid 10K Bitcoin for two pizzas.
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/8>**

## Шаг 9

### Курсы валют

Вследствие кибератаки на банк «Разбогатеем вместе» сломался алгоритм, выводящий курсы валют для определённой даты в мобильном приложении. Технический отдел банка просит вас исправить ситуацию и наладить вывод. На вход программе подаются следующие значения:

- дата (в формате ДД-ММ-ГГГГ)
- курс евро (сколько российских рублей стоит 1 евро)
- курс юаня (сколько российских рублей стоит 1 юань)

Напишите программу, которая выводит строку, показывающую, сколько российских рублей стоит 1 евро и 1 юань на указанную дату в формате:

```
На <дата>: 1€ = <курс евро>₽, 1¥ = <курс юаня>₽
```

#### Формат входных данных

На вход программе подаются три строки (каждая на отдельной строке): дата, курс евро и курс юаня.

#### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

#### ► Тестовые данные

##### Sample Input 1:

```
16-03-2024  
99.9718  
12.7328
```

##### Sample Output 1:

```
На 16-03-2024: 1€ = 99.9718₽, 1¥ = 12.7328₽
```

##### Sample Input 2:

```
12-05-2016  
75.4505  
10.1721
```

##### Sample Output 2:

```
На 12-05-2016: 1€ = 75.4505₽, 1¥ = 10.1721₽
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/9>

## Шаг 10

### Сумма кубов Куб суммы

Очень часто студенты "Поколения Python" путают понятия «сумма кубов» и «куб суммы». Для того чтобы внести ясность в этот извечный математический вопрос, предлагаем вам решить следующую задачу.

На вход программе подаются два целых числа  $a$  и  $b$ . Ваша программа должна посчитать для этих чисел сумму их кубов и куб их суммы и вывести результат вычислений в следующем формате:

Для чисел <число a> и <число b>:

Сумма кубов: <число a>\*\*3 + <число b>\*\*3 = <сумма кубов а и b>

Куб суммы: (<число a> + <число b>)\*\*3 = <куб суммы а и b>

## Формат входных данных

На вход программе подаются два целых числа  $a$  и  $b$ , каждое на отдельной строке.

## Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание.** Не перепутайте сумму кубов и куб суммы. 😊

► Тестовые данные

### Sample Input 1:

```
1
2
```

### Sample Output 1:

Для чисел 1 и 2:

Сумма кубов:  $1**3 + 2**3 = 9$

Куб суммы:  $(1 + 2)**3 = 27$

### Sample Input 2:

```
-2
0
```

### Sample Output 2:

Для чисел -2 и 0:

Сумма кубов:  $-2**3 + 0**3 = -8$

Куб суммы:  $(-2 + 0)**3 = -8$

**Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/10>**

## Шаг 11

### (Не) Активное похудение

Гвидо, засевший за компьютером и не ведущий активный образ жизни, «немного» поднабрал в весе. Осталось всего 60 дней до лета, а хочется быть в форме. Вот Гвидо и решился на похудение. Все дни до лета он пронумеровал от 1 до 60 (включительно). Перед началом похудения у Гвидо был вес 100 кг, а своей целью он поставил достичь веса 88 кг (или меньше). Он решил худеть на одну и ту же массу ежедневно.

Напишите программу, которая принимает на вход текущий день и текущий вес Гвидо. Программа должна вывести фразу:

- «Все идет по плану» (без кавычек), если Гвидо удаётся держать планку в похудении и его вес ниже либо равен тому, который он запланировал на текущий день
- «Что-то пошло не так» (без кавычек), если Гвидо не очень старается и его вес выше того, который он запланировал на текущий день

Также программа должна вывести информацию о номере дня похудения, текущем весе Гвидо и цели по весу на текущий день в формате:

```
#<номер дня> ДЕНЬ: ТЕКУЩИЙ ВЕС = <текущий вес Гвидо> кг, ЦЕЛЬ по ВЕСУ = <цель по весу на текущий день> кг
```

## Формат входных данных

На вход программе подаются два числа (каждое на отдельной строке): номер дня похудения (целое число) и текущий вес Гвида (действительное число).

## Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание.** В 1-й день похудения Гвидо уже должен похудеть (см. 1 тест).

► Подсказка

► Тестовые данные 

### Sample Input 1:

```
1  
99.9
```

### Sample Output 1:

```
Что-то пошло не так  
#1 ДЕНЬ: ТЕКУЩИЙ ВЕС = 99.9 кг, ЦЕЛЬ по ВЕСУ = 99.8 кг
```

### Sample Input 2:

```
1  
99.16
```

### Sample Output 2:

```
Все идет по плану  
#1 ДЕНЬ: ТЕКУЩИЙ ВЕС = 99.16 кг, ЦЕЛЬ по ВЕСУ = 99.8 кг
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1209103/step/11>

# 9.7 Строки в памяти компьютера, таблица символов Unicode

## Шаг 1

### Тема урока: представление строк в памяти компьютера, ASCII и Unicode

1. Представление строк в памяти компьютера
2. Таблица символов ASCII
3. Таблица символов Unicode
4. Функция `ord()`
5. Функция `chr()`

**Аннотация.** Представление строк в памяти компьютера.

#### Представление строк в памяти компьютера

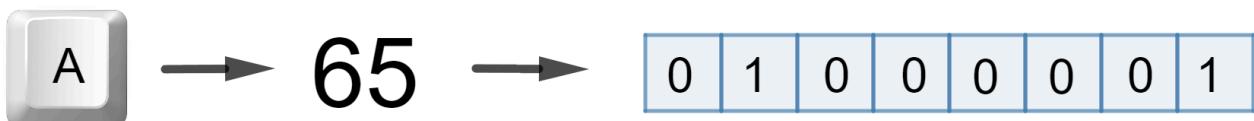
Любой набор данных в оперативной памяти

([https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F\\_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C)) компьютера должен храниться в виде двоичного числа. Это относится и к строкам, которые состоят из символов (буквы, знаки препинания и так далее). Когда символ сохраняется в памяти, он сначала преобразуется в цифровой код. И затем этот цифровой код сохраняется в памяти как двоичное число.

За прошедшие годы для представления символов в памяти компьютера были разработаны различные схемы кодирования. Исторически самой важной из этих схем кодирования является схема кодирования ASCII (American Standard Code for Information Interchange – американский стандартный код обмена информацией).

#### Таблица символов ASCII

ASCII представляет собой набор из 128 цифровых кодов, которые обозначают английские буквы, различные знаки препинания и другие символы. Например, код ASCII для прописной английской буквы «A» (латинской) равняется 65. Когда на компьютерной клавиатуре вы набираете букву «A» в верхнем регистре, в памяти сохраняется число 65 (как двоичное число, разумеется).



Код ASCII для английской «B» в верхнем регистре равняется 66, для «C» в верхнем регистре – 67 и так далее. **На один символ в ASCII отводится ровно 7 бит.**



Аббревиатура ASCII произносится «аски».

## ASCII table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Набор символов ASCII был разработан в начале 1960-х годов и в конечном счете принят почти всеми производителями компьютеров. Однако схема кодирования ASCII имеет ограничения, потому что она определяет коды только для 128 символов. Для того чтобы это исправить, в начале 1990-х годов был разработан набор символов Юникода (Unicode). Это широкая схема кодирования, совместимая с ASCII, которая может также представлять символы многих языков мира. Сегодня Юникод быстро становится стандартным набором символов, используемым в компьютерной индустрии.

## Таблица символов Unicode

Таблица символов Юникод представляет собой набор цифровых символов, которые включают в себя знаки почти всех письменных языков мира. Стандарт предложен в 1991 году некоммерческой организацией «Консорциум Юникода». Применение этого стандарта позволяет закодировать очень большое число символов из разных систем письменности: в документах, закодированных по стандарту Юникод, могут соседствовать китайские иероглифы, математические символы, буквы греческого алфавита, латиницы и кириллицы, символы музыкальной нотации.

Стандарт состоит из двух основных частей: универсального набора символов и семейства кодировок (Unicode transformation format, UTF). Универсальный набор символов перечисляет допустимые по стандарту Юникод символы и присваивает каждому символу код в виде неотрицательного целого числа. Семейство кодировок определяет способы преобразования кодов символов для хранения на компьютере и передачи.

В Юникод все время добавляются новые символы, а сам размер этой таблицы не ограничен и будет только расти, поэтому сейчас при хранении в памяти одного юникод-символа может потребоваться от 1 до 8 байт. Отсутствие ограничений привело к тому, что стали появляться символы на все случаи жизни.

 В Python строки хранятся в виде последовательности юникод символов.

## Примечания

**Примечание 1.** Официальный сайт таблицы символов Unicode (<https://home.unicode.org/>).

**Примечание 2.** Юникод – это не кодировка. Это именно таблица символов. То, как символы с соответствующими кодами будут храниться в памяти компьютера, зависит от конкретной кодировки, базирующейся на Юникоде, например UTF-8.

**Примечание 3.** Первые 128 кодов таблицы символов Unicode совпадают с ASCII.

## Шаг 2

### Функция `ord()`

Функция `ord()` позволяет определить код некоторого символа в таблице символов Unicode. Аргументом данной функции является одиночный символ.

Приведённый ниже код:

```
num1 = ord('A')
num2 = ord('B')
num3 = ord('a')
print(num1, num2, num3)
```

выводит:

```
65 66 97
```

Обратите внимание, что функция `ord()` принимает именно **одиночный символ**. Если попытаться передать строку, содержащую более одного символа, мы получим ошибку времени выполнения.

Приведённый ниже код:

```
num = ord('Abc')
print(num)
```

приводит к возникновению ошибки:

```
TypeError: ord() expected a character, but string of length 3 found
```



Название функции `ord()` происходит от английского слова «order» — порядок.

### Функция `chr()`

Функция `chr()` позволяет определить по коду символа сам символ. Аргументом данной функции является численный код.

Приведённый ниже код:

```
chr1 = chr(65)
chr2 = chr(75)
chr3 = chr(110)

print(chr1, chr2, chr3)
```

выводит:

```
A K n
```



Название функции `chr()` происходит от английского слова «character» — символ.

Функции `ord()` и `chr()` часто работают в паре. Мы можем использовать следующий код для вывода всех заглавных букв английского алфавита:

```
for i in range(26):
    print(chr(ord('A') + i))
```

Приведённый выше код выводит:

```
A  
B  
C  
...  
X  
Y  
Z
```

Вызов функции `ord('A')` возвращает код символа `A`, который равен 65. Далее на каждой итерации цикла к данному коду прибавляется значение переменной `i = 0, 1, 2, ..., 25`, а затем полученный код преобразуется в символ с помощью вызова функции `chr()`.

## Примечания

**Примечание 1.** Функции `ord()` и `chr()` являются **взаимно обратными**. Для них выполнены равенства:

$$\text{chr}(\text{ord}(<\text{символ}>)) = <\text{символ}>$$

$$\text{ord}(\text{chr}(<\text{код символа}>)) = <\text{код символа}>$$

Например, `ord('A')` возвращает число 65, а `chr(65)` возвращает символ 'A'. То есть `chr(ord('A'))` возвращает символ `A`. В обратную сторону это также работает: `ord(chr(65))` возвращает число 65.

Приведённый ниже код:

```
print(chr(ord('A')) == 'A')  
print(ord(chr(65)) == 65)
```

выводит:

```
True  
True
```

**Примечание 2.** Обратите внимание, что некоторые символы могут выглядеть одинаково, но на самом деле иметь **разные коды** в таблице Unicode. Это в первую очередь касается букв, которые имеют одинаковое написание на разных языках.

Приведённый ниже код:

```
print(ord('a'))      # английская буква «а»  
print(ord('а'))      # русская буква «а»
```

выводит:

```
97  
1072
```

Также можно заметить, что в таблице Unicode русские буквы находятся гораздо **дальше** английских.

 Happy Pythoning! 

## Шаг 3

Соотнесите символы с их кодами в таблице Unicode (ASCII).

**Примечание 1.** Гарантируется, что все буквенные символы в данной задаче являются буквами английского алфавита.

**Примечание 2.** Чтобы узнать код символа в таблице Unicode (ASCII), можно воспользоваться функцией `ord()`.

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/3>

## Шаг 4

Что покажет приведённый ниже код?

```
print(chr(ord('Ѐ')))  
print(ord(chr(128013)))
```

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/4>

## Шаг 5

Что покажет приведённый ниже код?

```
print(ord('foo'))
```

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/5>

## Шаг 6

### Какая следующая буква? SOON

На вход программе подаётся некоторая буква русского алфавита в верхнем регистре. Найдите следующую за ней букву и выведите её на экран. Если введённая буква является последней в алфавите, то выведите текст «Дальше букв нет» (без кавычек).

#### Формат входных данных

На вход программе подаётся одна буква русского алфавита в верхнем регистре.

#### Формат выходных данных

Программа должна вывести одну букву в верхнем регистре или текст «Дальше букв нет» (без кавычек) в соответствии с условием задачи.

**Примечание.** Будем считать, что буквы Ё нет в русском алфавите. 

▶ Тестовые данные 

**Sample Input 1:**

А

**Sample Output 1:**

Б

**Sample Input 2:**

Я

**Sample Output 2:**

Дальше букв нет

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/6>

**Шаг 7**

## Символы в диапазоне

На вход программе подаются два числа  $a$  и  $b$  ( $a < b$ ). Напишите программу, которая для каждого кодового значения в диапазоне от  $a$  до  $b$  (включительно) выводит соответствующий ему символ из таблицы символов Unicode.

**Формат входных данных**

На вход программе подаются два натуральных числа, каждое на отдельной строке.

**Формат выходных данных**

Программа должна вывести текст в соответствии с условием задачи.

► Тестовые данные

**Sample Input 1:**

```
65  
70
```

**Sample Output 1:**

```
A B C D E F
```

**Sample Input 2:**

```
97  
110
```

**Sample Output 2:**

```
a b c d e f g h i j k l m n
```

**Sample Input 3:**

```
48  
64
```

**Sample Output 3:**

```
0 1 2 3 4 5 6 7 8 9 : ; < = > ? @
```

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/7>

**Шаг 8**

## Простой шифр

На вход программе подаётся строка текста. Напишите программу, которая переводит каждый ее символ в соответствующий ему код из таблицы символов Unicode.

## Формат входных данных

На вход программе подаётся строка текста.

## Формат выходных данных

Программа должна вывести кодовые значения символов строки разделенных одним символом пробела.

**Примечание.** Проверяющая система примет ваше решение, даже если в конце будет лишний пробел. 😊

► Тестовые данные

### Sample Input:

```
Hello world!
```

### Sample Output:

```
72 101 108 108 111 32 119 111 114 108 100 33
```

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/8>

## Шаг 9

## Самое тяжёлое слово



Под "тяжестью" слова будем понимать сумму кодов по таблице Unicode всех символов этого слова. Напишите программу, которая принимает 4 слова и находит среди них **самое тяжёлое** слово. Если самых тяжёлых слов будет несколько, то программа должна вывести первое из них.

## Формат входных данных

На вход программе подаются 4 слова, каждое на отдельной строке.

## Формат выходных данных

Программа должна вывести самое тяжёлое слово в строке.

► Тестовые данные

### Sample Input 1:

```
строки
списки
кортежи
множества
```

### Sample Output 1:

```
множества
```

### Sample Input 2:

```
az
by
cx
122
```

### Sample Output 2:

```
az
```

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/9>

## Шаг 10

### Стоимость ответа

Модератору Сэму за **каждый символ** его сообщений в комментариях Тимур платит в 🐝 (пчёлках-coin) по следующему тарифу:

<код символа в таблице Unicode> × 3 🐝

А стоимость всего сообщения складывается из суммы стоимостей всех символов. Сэму захотелось подсчитать, сколько 🐝 он зарабатывает за свои ответы в комментариях, и просит вас помочь ему.

На вход программе подаётся строка текста. Требуется написать программу, которая находит стоимость сообщения Сэма в 🐝 и выводит текст в следующем формате:

```
Текст сообщения: '<текст сообщения Сэма>'  
Стоимость сообщения: <стоимость сообщения Сэма> 🐝
```

#### Формат входных данных

На вход программе подаётся строка текста – очередной ответ Сэма в комментариях.

#### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание.** 🐝 (пчёлка-coin) – виртуальная валюта команды BEEGEEK, которой Тимур расплачивается со своими сотрудниками.

#### ▶ Тестовые данные

##### Sample Input 1:

```
@кодер 666, пишите в комментариях по делу, не засоряйте чат бредом
```

##### Sample Output 1:

```
Текст сообщения: '@кодер 666, пишите в комментариях по делу, не засоряйте чат бредом'  
Стоимость сообщения: 164457 🐝
```

##### Sample Input 2:

```
@тот самый Гвидо, у вас программа выводит лишний пробел в конце первой строки
```

##### Sample Output 2:

```
Текст сообщения: '@тот самый Гвидо, у вас программа выводит лишний пробел в конце первой строки'  
Стоимость сообщения: 206064 🐝
```

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/10>

## Шаг 11

### Накручиваем стоимость ответа

Как вы помните из прошлой задачи, модератору Сэму за **каждый символ** его сообщений в комментариях Тимур платит в 🐝 (пчёлках-coin) по следующему тарифу:

<код символа в таблице Unicode> × 3 🐝

А стоимость всего сообщения складывается из суммы стоимостей всех символов. На этот раз Сэму захотелось схитрить и попробовать увеличить стоимость своего сообщения, заменив в нем некоторые английские буквы на русские. Как вы помните, русские буквы в таблице Unicode находятся после английских.

Сэм хочет заменить следующие **английские** буквы:

еуорахсЕТОРАНХСВМ

на соответствующие им **русские** буквы:

еуорахсЕТОРАНХСВМ

Тимур визуально разницу не заметит, а Сэм сможет зарабатывать больше пчёлок-coin. 😊

На вход программе подаётся строка текста. Требуется написать программу, которая находит стоимость старого и нового сообщений Сэма в 🐝 и выводит текст в следующем формате:

Старая стоимость: <стоимость старого сообщения> 🐝

Новая стоимость: <стоимость нового сообщения> 🐝

#### Формат входных данных

На вход программе подаётся строка текста – очередной ответ Сэма в комментариях.

#### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание.** Обратите внимание, что если в строке не удаётся заменить буквы, то стоимость сообщения не изменится. 🤔

#### ► Тестовые данные

#### Sample Input 1:

@coder666, пишите в комментариях по делу, не засоряйте чат бредом

#### Sample Output 1:

Старая стоимость: 149709 🐝

Новая стоимость: 158532 🐝

#### Sample Input 2:

@Timur\_Guiev, ХОЧУ БОЛЬШЕ ПЧЕЛОК, ПЧЕЛОК, ПЧЕЛОК

#### Sample Output 2:

Старая стоимость: 137946 🐝

Новая стоимость: 137946 🐝

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/11>

## Шаг 12

### Шифр Цезаря

Легион Цезаря, созданный в 23 веке на основе Римской Империи не изменяет древним традициям и использует шифр Цезаря ([https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80\\_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F)). Это их и подвело, ведь данный шифр очень простой. Однако в постапокалипсисе люди плохо знают все тонкости довоенного мира, поэтому ученые из НКР не могут понять, как именно нужно декодировать данные сообщения. Напишите программу для декодирования этого шифра.

#### Формат входных данных

В первой строке подаётся число  $n$  ( $1 \leq n \leq 25$ ) – сдвиг, во второй строке даётся закодированное сообщение в виде строки со строчными латинскими буквами.

#### Формат выходных данных

Программа должна вывести одну строку – декодированное сообщение. Обратите внимание, что нужно декодировать сообщение, а не закодировать.

► Тестовые данные 

#### Sample Input 1:

```
1  
bwfusvfupdbftbs
```

#### Sample Output 1:

```
avetruetocaesar
```

#### Sample Input 2:

```
14  
fsfftsfufksttskskt
```

#### Sample Output 2:

```
rerrfergrweffewewf
```

**Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/12>**

## Шаг 13

### Сбой в системе

После недавнего сбоя в операционной системе от компании «Oursoft» у Гвидо сбилась кодировка на компьютере. Теперь все буквы **русского алфавита** отображаются в некорректном виде:

```
[u-<номер символа в таблице Unicode>]
```

Гвидо ещё не научился читать символы в таком формате, поэтому просит вас написать программу, которая будет "расшифровывать" для него все тексты на компьютере.

На вход программе подаётся строка текста. Расшифруйте текст, заменив все конструкции 

```
[u-<номер символа в таблице Unicode>]
```

 на соответствующие буквы русского алфавита, и выведите его.

#### Формат входных данных

На вход программе подаётся строка текста, в которой могут быть зашифрованы символы русского алфавита.

## Формат выходных данных

Программа должна вывести строку текста, расшифровав символы русского алфавита.

**Примечание.** Будем считать, что буквы Ё нет в русском алфавите. 🤪

► Тестовые данные ●

### Sample Input 1:

```
Hello, my name is [u-1061][u-1072][u-1082][u-1080]!
```

### Sample Output 1:

```
Hello, my name is Хаки!
```

### Sample Input 2:

```
Username: [u-1042][u-1072][u-1089][u-1103]; City: [u-1050][u-1072][u-1079][u-1072][u-1085][u-1100]
```

### Sample Output 2:

```
Username: Вася; City: Казань
```

### Sample Input 3:

```
Because I didn't know what I had until it was gone! All right?
```

### Sample Output 3:

```
Because I didn't know what I had until it was gone! All right?
```

Чтобы решить это задание откройте <https://stepik.org/lesson/313439/step/13>

## 9.8 Сравнение строк

### Шаг 1

#### Тема урока: сравнение строк

1. Сравнение строк единичной длины
2. Сравнение строк не единичной длины

**Аннотация.** Урок посвящён сравнению строк: алгоритму и способам сравнения.

В Python мы можем сравнивать с помощью операторов `==`, `!=`, `<`, `<=`, `>` и `>=` не только числа, но и строки. В отличие от чисел, сравнение строк происходит на основе **лексикографического порядка** – в соответствии с кодами составляющих их символов в таблице Unicode.

#### Сравнение строк единичной длины

Начнем с примера сравнения строк, состоящих из одного символа. В Python это сравнение происходит путём сравнения кодов этих символов в таблице Unicode.

Приведённый ниже код:

```
print('a' > 'b')
print('a' < 'z')
```

выводит:

```
False
True
```

Действительно, код символа `a` в таблице Unicode равен числу 97, а символа `b` – числу 98. Число 97 меньше числа 98, поэтому и символ `a` меньше символа `b`. Аналогично символ `z` больше символа `a`, потому что код символа `z` (число 122) больше кода символа `a` (число 97).

Предыдущий код полностью эквивалентен следующему коду:

```
print(ord('a') > ord('b'))
print(ord('a') < ord('z'))
```

Обратите внимание, что буквы в нижнем регистре всегда больше своих аналогов в верхнем регистре. Для букв русского алфавита это правило также работает.

Приведённый ниже код:

```
print('д' > 'Д')
print('ы' < 'Ы')
```

выводит:

```
True
True
```

Ничего удивительного в этом нет, потому что буквы в нижнем регистре идут после букв в верхнем регистре в таблице Unicode.



Подробнее ознакомиться с таблицей символов Unicode можно по [ссылке](https://symbl.cc/en/unicode-table/) (<https://symbl.cc/en/unicode-table/>). Обычно на практике достаточно оперировать таблицей ASCII (<https://www.asciitable.com/>), которая является подмножеством таблицы Unicode. Первые 128 символов таблицы Unicode совпадают с таблицей ASCII.

## Сравнение строк не единичной длины

Обычно мы работаем не с отдельными символами, а со строками, которые состоят из нескольких символов сразу.

**Алгоритм сравнения строк:**

1. Начинаем с первых символов каждой строки. Если символы равны, переходим к следующей паре символов
2. Когда находим первый отличающийся символ, строка с меньшим символом считается "меньше"
3. Если одна из строк заканчивается раньше, то более короткая считается "меньше"

Чтобы лучше разобраться с алгоритмом сравнения строк, рассмотрим несколько примеров.

**Пример 1.** Сравним строки `'hello'` и `'hell'`.

- Сравнение первых символов: `h` и `h` – оба символа равны, переходим к следующей паре символов
- Сравнение вторых символов: `e` и `e` – оба символа равны, переходим к следующей паре символов
- Сравнение третьих символов: `l` и `l` – оба символа равны, переходим к следующей паре символов
- Сравнение четвертых символов: `l` и `l` – оба символа равны, переходим к следующей паре символов
- Сравнение пятых символов: `o` (у первой строки) и отсутствующий символ (у второй строки) – вторая строка закончилась

Поскольку у второй строки закончились символы, а у первой строки они еще есть, считается, что первая строка больше второй. Поэтому `'hello' > 'hell'`.

h e l l o  
h e l l

**Пример 2.** Сравним строки `'men'` и `'mya'`.

- 1) Сравнение первых символов: `m` и `m` – оба символа равны, переходим к следующей паре символов
- 2) Сравнение вторых символов: `e` (у первой строки) и `y` (у второй строки) – символ `e` (число 101) меньше символа `y` (число 121)

Так как строки начинают различаться со второго символа, то на основе этого символа и делается вывод о результатах сравнения строк. В данном случае символ `e` меньше символа `y`. Получаем, что `'men' < 'mya'`.

m e n  
m y a



Обратите внимание, что в Python сравнение останавливается, как только находится первое различие между символами на соответствующих позициях. Дальнейшее сравнение символов не требуется.

**Пример 3.** Сравним строки `'Meeeeooooooow'` и `'meow'`.

- 1) Сравнение первых символов: `M` (у первой строки) и `m` (у второй строки) – символ `M` (число 77) меньше символа `m` (число 109).

Строки различаются уже на первых символах, и первый символ у первой строки меньше первого символа второй строки. Поэтому `'Meeeeooooooow' < 'meow'`.

# Meeeeooooow meow

## Примечания

**Примечание 1.** Нельзя путать сравнение чисел и сравнение строк, содержащих эти числа.

Приведённый ниже код:

```
print(10 > 9)
print('10' > '9')
```

выводит:

```
True
False
```

**Примечание 2.** Мы можем сравнивать не только строки, состоящие из букв латинского алфавита, но и строки, состоящие из любых символов, которые входят в таблицу Unicode. Алгоритм сравнения строк при этом будет аналогичный – в соответствии с кодами символов в таблице Unicode.

Приведённый ниже код:

```
print('Тинькофф' == 'Т-банк')
print('$' > '$€')
print(max('😊', '☕', '⟲'))
```

выводит:

```
False
True
⟲
```

**Примечание 3.** В Python не поддерживается операция сравнения строк и чисел друг с другом.

Приведённый ниже код:

```
print('45' > 44)
```

приводит к возникновению ошибки:

```
TypeError: '>' not supported between instances of 'str' and 'int'
```

Python пытается выполнить лексикографическое сравнение для строк и числовое сравнение для чисел. Но эти операции несопоставимы, и поэтому возникает ошибка.

**Примечание 4.** В Python встроенные функции `min()` и `max()` могут принимать строки в качестве аргументов и сравнивают их лексикографически (используя порядок символов в кодировке Unicode). Как несложно догадаться, функция `min()` вернёт самую "маленькую" строку, а `max()` – самую "большую" строку.

Приведённый ниже код:

```
print(max('tree', 'try', 'true'))
print(min('cat', 'car', 'cape'))
```

выводит:

```
try  
    cape
```

Обратите внимание, что мы не можем **одновременно** передавать строки и числа в качестве аргументов в функции `min()` и `max()`. Это является следствием того, что мы не можем сравнивать строки с числами.

Приведённый ниже код:

```
print(min('2', 8, '45', 90))
```

приводит к возникновению ошибки:

```
TypeError: '<' not supported between instances of 'int' and 'str'
```



## Шаг 2

Определите, какой из знаков (`>`, `<`, `==`) нужно поставить на место знака вопроса (`?`), чтобы выражение оказалось истинным.

**Примечание.** Гарантируется, что все буквенные символы в данной задаче являются буквами английского алфавита.

**Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/2>**

## Шаг 3

Определите, какой из знаков (`>`, `<`, `==`) нужно поставить на место знака вопроса (`?`), чтобы выражение оказалось истинным.

**Примечание.** Гарантируется, что все буквенные символы в данной задаче являются буквами английского алфавита.

**Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/3>**

## Шаг 4

Расположите строки сверху вниз в порядке возрастания.

**Примечание 1.** Самые "большие" строки должны быть в самом низу, а самые "маленькие" – в самом верху.

**Примечание 2.** Гарантируется, что все буквенные символы в данной задаче являются буквами английского алфавита.

**Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/4>**

## Шаг 5

Расположите строки сверху вниз в порядке возрастания.

**Примечание.** Самые "большие" строки должны быть в самом низу, а самые "маленькие" – в самом верху.

**Чтобы решить это задание откройте** <https://stepik.org/lesson/1402735/step/5>

## Шаг 6

Расположите строки сверху вниз в порядке возрастания.

**Примечание 1.** Самые "большие" строки должны быть в самом низу, а самые "маленькие" – в самом верху.

**Примечание 2.** Гарантируется, что все буквенные символы в данной задаче являются буквами английского алфавита.

**Чтобы решить это задание откройте** <https://stepik.org/lesson/1402735/step/6>

## Шаг 7

Что покажет приведённый ниже код?

```
print(max(9, 10, 11))
print(max('9', '10', '11'))
```

**Чтобы решить это задание откройте** <https://stepik.org/lesson/1402735/step/7>

## Шаг 8

Что покажет приведённый ниже код?

```
print(min('9', '10', '11') + max('9', '10', '11'))
```

**Чтобы решить это задание откройте** <https://stepik.org/lesson/1402735/step/8>

## Шаг 9

Что покажет приведённый ниже код?

```
print(min(10, 5, 15) + max('10', '5', '15'))
```

**Чтобы решить это задание откройте** <https://stepik.org/lesson/1402735/step/9>

## Шаг 10

Что покажет приведённый ниже код?

```
print(min(10, 5, 15) + max(10, 5, '15'))
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/10>

## Шаг 11

### Строковые минимум и максимум

На вход программе подаётся последовательность строк, каждая строка на отдельной строке. Концом последовательности является слово «КОНЕЦ» (без кавычек). При этом само слово «КОНЕЦ» не входит в последовательность, лишь символизируя ее окончание. Напишите программу, которая находит в данной последовательности **максимальную и минимальную** строки (в лексикографическом порядке) и выводит их в следующем формате:

Минимальная строка : <минимальная строка>  
Максимальная строка : <максимальная строка>

#### Формат входных данных

На вход программе подаётся последовательность строк, каждая на отдельной строке.

#### Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

**Примечание.** Не только у чисел мы можем находить максимум и минимум. 😊

► Тестовые данные

#### Sample Input 1:

```
буря
мглою
небо
кроет
КОНЕЦ
```

#### Sample Output 1:

Минимальная строка : буря  
Максимальная строка : небо

#### Sample Input 2:

```
вихри
снежные
крутя
КОНЕЦ
```

#### Sample Output 2:

Минимальная строка : вихри  
Максимальная строка : снежные

Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/11>

## Шаг 12

### Волшебное число

В некотором наборе слов Сэм находит "волшебное" число по следующему алгоритму: берет самую "маленькую" и самую "большую" строки, перемножает Unicode-коды последних символов этих строк и возводит полученное число в квадрат. Результатом и является "волшебное" число.

На вход программе подаются 4 слова. Найдите "волшебное" число в этом наборе слов.

#### Формат входных данных

На вход программе подаются 4 слова, каждое на отдельной строке.

#### Формат выходных данных

Программа должна вывести "волшебное" число в наборе слов.

► Тестовые данные 

#### Sample Input 1:

```
I  
will  
be  
back
```

#### Sample Output 1:

```
62157456
```

#### Sample Input 2:

```
all  
dreams  
come  
true
```

#### Sample Output 2:

```
118984464
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/12>

## Шаг 13

### Название класса

В школе BEEGEEK названия учебных классов необычные. Они имеют следующий формат:

```
<номер класса><буква класса>
```

где <номер класса> должен находиться в диапазоне от 0 (как и все у программистов) до 9 включительно, а буквой класса могут быть все буквы в диапазоне от «А» до «П» включительно.

Напишите программу, которая принимает натуральное число  $n$ , а далее  $n$  названий классов, каждое на новой строке. Для каждого названия класса ваша программа должна выводить на отдельной строке «YES» (без кавычек), если название класса корректное, или «NO» (без кавычек) в противном случае.

#### Формат входных данных

На вход программе подаются натуральное число  $n$ , а затем  $n$  названий классов, каждое на отдельной строке.

#### Формат выходных данных

Программа должна вывести на отдельной строке для каждого названия класса «YES» (без кавычек) или «NO» (без кавычек) в соответствии с условием задачи.

**Примечание.** Будем считать, что буквы Ё нет в русском алфавите, а значит, класс с такой буквой также будет считаться некорректным. 😳

#### ► Тестовые данные

##### Sample Input 1:

```
5
9А
11Б
0К
5П
2У
```

##### Sample Output 1:

```
YES
NO
YES
YES
NO
```

##### Sample Input 2:

```
4
10А
00
5Д
2Я
```

##### Sample Output 2:

```
NO
YES
YES
NO
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/13>

#### Шаг 14

## Необычное сравнение



На вход программе подаются 2 строки. Вам необходимо сравнить эти строки посимвольно, не учитывая регистр и игнорируя все небуквенные символы. Программа должна вывести «YES» (без кавычек), если строки окажутся равны в результате такой проверки, или «NO» (без кавычек) в противном случае.

## Формат входных данных

На вход программе подаются 2 строки, каждая на отдельной строке.

## Формат выходных данных

Программа должна вывести «YES» (без кавычек) или «NO» (без кавычек) в соответствии с условием задачи.

**Примечание.** Разберём 1-й тест:

A b ~~4~~ c ~~1\$#~~ d d d  
a ~~b~~ ~~c~~ ~~#~~ D D D ~~70~~

► Тестовые данные 

### Sample Input 1:

```
Ab4c1$ddd  
a_b_c##DDD70
```

### Sample Output 1:

```
YES
```

### Sample Input 2:

```
n5#e6vER  
+NEV-er
```

### Sample Output 2:

```
YES
```

### Sample Input 3:

```
C - Э - М  
C - Э - Э - М
```

### Sample Output 3:

```
NO
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/14>

## Шаг 15

### Сортируем слова

На вход программе подаются 3 различных слова. Вам необходимо отсортировать эти слова по возрастанию в лексикографическом порядке и вывести их на одной строке, разделяя символом пробела.

## Формат входных данных

На вход программе подаются 3 слова, каждое на отдельной строке.

## Формат выходных данных

Программа должна вывести 3 слова на одной строке, разделяя их символом пробела.

► Тестовые данные 

### Sample Input 1:

```
python  
java  
kotlin
```

### Sample Output 1:

```
java kotlin python
```

### Sample Input 2:

```
первое  
второе  
третье
```

### Sample Output 2:

```
второе первое третье
```

### Sample Input 3:

```
июнь  
июль  
август
```

### Sample Output 3:

```
август июль июнь
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/15>

## Шаг 16

### Порядок книг

Все книги в домашней библиотеке Душнилы, друга Сэма, должны быть обязательно отсортированы **по возрастанию**: сначала по фамилиям авторов, а в случае совпадения фамилий – по названиям. Напишите программу, которая проверяет, верно ли отсортированы книги.

На вход вашей программе поступает число  $n$ , а затем –  $n$  строк, каждая строка представляет собой книгу в следующем формате:

```
<фамилия автора> <инициалы автора>, «<название книги>»
```

Программа должна вывести «YES» (без кавычек), если книги отсортированы в соответствии с пожеланиями Душнилы, или «NO» (без кавычек) в противном случае.

#### Формат входных данных

На вход программе подаются натуральное число  $n$ , а затем –  $n$  строк.

#### Формат выходных данных

Программа должна вывести «YES» (без кавычек) или «NO» (без кавычек) в соответствии с условием задачи.

**Примечание 1.** Обратите внимание, что Душнила **игнорирует** инициалы автора при сортировке книг.

**Примечание 2.** Гарантируется, что книги в наборе не повторяются.

**Примечание 3.** Гарантируется, что фамилия автора состоит из одного слова.

► Тестовые данные

**Sample Input 1:**

```
5
Гоголь Н.В., «Мертвые души»
Гончаров И.А., «Обломов»
Пушкин А.С., «Капитанская дочка»
Тургенев И.С., «Ася»
Тургенев И.С., «Первая любовь»
```

**Sample Output 1:**

```
YES
```

**Sample Input 2:**

```
3
Толстой А.Н., «Петр Первый»
Толстой А.Н., «Хмурое утро»
Толстой Л.Н., «Война и мир»
```

**Sample Output 2:**

```
NO
```

Чтобы решить это задание откройте <https://stepik.org/lesson/1402735/step/16>