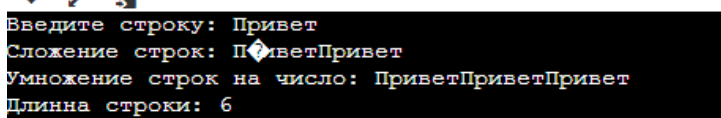


Строки

Строка считывается со стандартного ввода функцией `input()`. Напомним, что для двух строк определена операция сложения (конкатенации), также определена операция умножения строки на число.


Строка состоит из последовательности символов. Узнать количество символов (длину строки) можно при помощи функции `len`.

```
1 q = input('Введите строку: ')
2 print('Сложение строк:', q+q)
3 print('Умножение строк на число:', q*3)
4 print('Длина строки:', len(q))
```



Любой другой объект в Питоне можно перевести к строке, которая ему соответствует. Для этого нужно вызвать функцию `str()`, передав ей в качестве параметра объект, переводимый в строку.

```
1 # Это число
2 numb = 4
3 print('Сложение чисел:', numb+numb)
4 print('Умножение чисел:', numb*3)
5 # переводим число в строку, теперь numb - строка
6 numb = str(numb)
7 print('Сложение:', numb+numb)
8 print('Умножение:', numb*3)
```



Срезы

Срез (slice) — извлечение из данной строки одного символа или некоторого фрагмента подстроки или подпоследовательности.

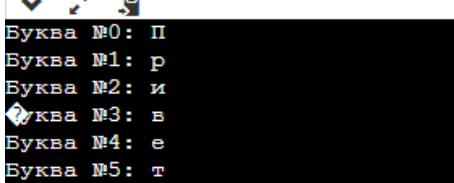
Номера символов в строке (а также в других структурах данных: списках, кортежах) называются **индексом**.

Есть три формы срезов.

Первая форма: срез с одним параметром.

Самая простая форма среза: **взятие одного символа строки**, а именно, `S[i]` — это срез, состоящий из одного символа, который имеет номер `i`. При этом считается, что **нумерация начинается с числа 0**. То есть если `S = 'Привет'`, то `S[0] == 'П'`, `S[1] == 'р'`, `S[2] == 'и'`, `S[3] == 'в'`, `S[4] == 'е'`, `S[5] == 'т'`.

```
1 S = 'Привет'
2 print('Буква №0:', S[0])
3 print('Буква №1:', S[1])
4 print('Буква №2:', S[2])
5 print('Буква №3:', S[3])
6 print('Буква №4:', S[4])
7 print('Буква №5:', S[5])
```



Если указать отрицательное значение индекса, то номер будет отсчитываться с конца, начиная с номера -1. То есть `S[-1] == 'т'`, `S[-2] == 'е'`, `S[-3] == 'в'`, `S[-4] == 'и'`, `S[-5] == 'р'`, `S[-6] == 'П'`

```
1 S = 'Привет'
2 print('Буква № -1:', S[-1])
3 print('Буква № -2:', S[-2])
4 print('Буква № -3:', S[-3])
5 print('Буква № -4:', S[-4])
6 print('Буква № -5:', S[-5])
7 print('Буква № -6:', S[-6])
```

Буква № -1: т
Буква № -2: е
Буква № -3: в
Буква № -4: и
Буква № -5: р
Буква № -6: П

Если же номер символа в срезе строки `S` больше либо равен `len(S)`, или меньше, чем `-len(S)`, то при обращении к этому символу строки произойдет ошибка **`IndexError: string index out of range`**

```
1 S = 'Привет'
2 print('Буква № 45:', S[45])
```

Traceback (most recent call last):
File "main.py", line 2, in <module>
print('Буква № 45:', S[45])
IndexError: string index out of range

```
1 S = 'Привет'
2 print('Буква № -10:', S[-10])
```

Traceback (most recent call last):
File "main.py", line 2, in <module>
print('Буква № -10:', S[-10])
IndexError: string index out of range

Вторая форма срезов: срез с двумя параметрами.

Срез с двумя параметрами: `S[a:b]` возвращает подстроку из `b - a` символов, начиная с символа с индексом `a`, то есть до символа с индексом `b`, **не включая его**.

Например, `S[1:4] == 'риве'`, то же самое получится если написать `S[-4:-1]`.

```
1 S = 'Привет'
2 print('Положительный срез:', S[1:5])
3 print('Отрицательный срез:', S[-5:-1])
```

Положительный срез: риве
Отрицательный срез: риве

Можно использовать как положительные, так и отрицательные индексы в одном срезе, например, `S[1:-1]` — это строка без первого и последнего символа (срез начинается с символа с индексом 1 и заканчивается индексом -1, не включая его).

```
1 S = 'Привет'
2 print('Смешанный срез:', S[1:-1])
```

Смешанный срез: риве

Если опустить второй параметр (но поставить двоеточие), то срез берется до конца строки. Например, чтобы удалить из строки первый символ (его индекс равен 0), можно взять срез `S[1:]`. Аналогично если опустить первый параметр, то можно взять срез от начала строки. То есть удалить из строки последний символ можно при помощи среза `S[:-1]`. Срез `S[:]` совпадает с самой строкой `S`.

```
1 S = 'Привет'
2 print('Срез без второго параметра:', S[1:])
3 print('Срез без первого параметра:', S[:-1])
4 print('Срез без параметров:', S[:])
```

Срез без второго параметра: ривет
Срез без первого параметра: Приве
Срез без параметров: Привет

Третья форма срезов: срез с тремя параметрами.

Если задать срез с тремя параметрами `S[a:b:d]`, то третий параметр задает шаг, как в случае с функцией `range`, то есть будут взяты символы с индексами `a`, `a + d`, `a + 2 * d` и т. д. При задании значения третьего параметра, равному 2, в срез попадет каждый второй символ, а если взять значение среза, равное -1, то символы будут идти в обратном порядке. Например, можно перевернуть строку срезом `S[::-1]`.

```
1 S = 'Привет мир, я здесь'
2 print('Каждый второй символ с начала:', S[::2])
3 print('Каждый второй символ с конца:', S[::-2])
4 print('Переворот строки:', S[::-1])
```

Каждый второй символ с начала: Пие язь
Каждый второй символ с конца: ьезя,и ейП
Переворот строки: ьседз я ,рим тевиP

Далее приведены примеры использования срезов с различным количеством параметров

```
1 s = 'АБВГДЕЖ'
2 print(s[1])      Б
3 print(s[-1])     Ж
4 print(s[1:3])    БВ
5 print(s[1:-1])   БВГДЕ
6 print(s[:3])     АБВ
7 print(s[2:])     ВГДЕЖ
8 print(s[: -1])   АБВГДЕ
9 print(s[::2])    АДЖ
10 print(s[1::2])  БГЕ
11 print(s[::-1])  ЖЕДГВБА
```

Обратите внимание на то, как похож третий параметр среза на третий параметр функции `range()`

```
1 s = 'АБВГДЕЖЗИКЛМНОПРСТУФ'
2 print(s[0:10:2])
3 for i in range(0, 10, 2):
4     print(i, s[i], end='; ')
```

АВДЖИ
0 А; 2 В; 4 Д; 6 Ж; 8 И;

Для того, чтобы посимвольно перебрать строку, можно использовать цикл `for` следующим образом:

```
1 S = 'АБВГДЕЖЗИКЛМНОПРСТУФ'
2 for i in range(0, len(S)):
3     print(S[i], end='; ')
```

А;Б;В;Г;Д;Е;Ж;З;И;К;Л;М;Н;О;П;Р;С;Т;У;Ф;

Методы

Метод — это 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"`.

Find


Метод `find` находит в данной строке (к которой применяется метод) данную подстроку (которая передается в качестве параметра). Функция возвращает **индекс первого вхождения искомой подстроки**. Если же подстрока не найдена, то метод возвращает значение -1.

```
1 S = 'Привет'
2 print(S.find('p'))
3 print(S.find('ив'))
4 print(S.find('ж'))
```

1
2
-1

Аналогично, метод **rfind** возвращает индекс последнего вхождения данной строки (“поиск справа”)

```
1 S = 'Привееет'
2 print(S.find('e'))
3 print(S.rfind('e'))
```




Если вызвать метод **find** с тремя параметрами **S.find(T, a, b)**, то поиск будет осуществляться в срезе **S[a:b]**. Если указать только два параметра **S.find(T, a)**, то поиск будет осуществляться в срезе **S[a:]**, то есть начиная с символа с индексом **a** и до конца строки. Метод **S.find(T, a, b)** возвращает индекс в строке **S**, а не индекс относительно среза.

Replace


Метод **replace** заменяет все вхождения одной строки на другую. Формат: **S.replace(old, new)** — заменить в строке **S** все вхождения подстроки **old** на подстроку **new**. Пример:

```
1 S = 'мама'
2 print(S.replace('м', 'п'))
```



Если методу **replace** задать еще один параметр: **S.replace(old, new, count)**, то заменены будут не все вхождения, а только не больше, чем первые **count** из них:


```
1 S = 'мама'
2 print(S.replace('м', 'п', 1))
```



Count

Метод **count** подсчитывает количество вхождений одной строки в другую строку. Простейшая форма вызова **S.count(T)** возвращает число вхождений строки **T** внутри строки **S**. При этом подсчитываются только непересекающиеся вхождения, например:

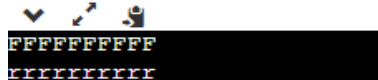
```
1 print('Абракадабра'.count('a'))
2 print('AAAAAAAAAA'.count('AA'))
```



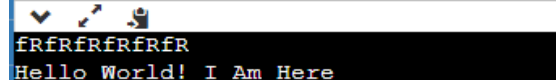
Upper, lower, swapcase, title

Методы **upper** и **lower** позволяют приводить строку к верхнему и нижнему регистру соответственно. Метод **swapcase** переводит символы нижнего регистра в верхний, а верхнего — в нижний. Метод **title** первую букву каждого слова переводит в верхний регистр, а все остальные — в нижний.

```
1 s = 'fffffffffff'
2 s1 = 'RRRRRRRRRR'
3 print(s.upper())
4 print(s1.lower())
```



```
1 s = 'FrFrFrFrFrFr'
2 s1 = 'heLLo wORLD! i aM hErE'
3 print(s.swapcase())
4 print(s1.title())
```



Задачи

№1. Работа со срезами

Дана строка.

Сначала выведите третий символ этой строки.

Во второй строке выведите предпоследний символ этой строки.

В третьей строке выведите первые пять символов этой строки.

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

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

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

В седьмой строке выведите все символы в обратном порядке.

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

В девятой строке выведите длину данной строки.

№2. Посчитать количество слов

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

№3. Переставить два слова местами

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

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

№4. Удаление символа

На вход подаётся две строки `S` и `N`: первая – строка `S`, в которой необходимо произвести удаление, вторая – символ `N`, который надо убрать из строки.

Пример: `S = 'Мама папа я'`, `N = 'а'`. Вывести программа должна следующую строку: `'Мм пп я'`

№5. Разделение строки пополам

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

№6. Сокращение имени и отчества

На вход подаётся строка, содержащая фамилию имя и отчество человека в полной форме. Вывести полученную информацию в формате Фамилия И.О.

Пример: Бачанцев Иван Владимирович преобразуется в Бачанцев И.В.

№7. Дополнительная (удаление каждого третьего символа)

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