* [ОБЗОР КУРСА](https://lyceum.yandex.ru/courses/123/groups/568)

[Урок Индексация](https://lyceum.yandex.ru/courses/123/groups/568/lessons/639)

**Строки. Индексация**

**План урока**

1

[Строка как коллекция](https://lyceum.yandex.ru/courses/123/groups/568/lessons/639/materials/964#1)

2

[Что мы знаем о строках](https://lyceum.yandex.ru/courses/123/groups/568/lessons/639/materials/964#2)

3

[Индексация в строках](https://lyceum.yandex.ru/courses/123/groups/568/lessons/639/materials/964#3)

4

[Перебор элементов строки](https://lyceum.yandex.ru/courses/123/groups/568/lessons/639/materials/964#4)

5

[Хранение текстов в памяти компьютера](https://lyceum.yandex.ru/courses/123/groups/568/lessons/639/materials/964#5)

**Аннотация**

*На этом занятии мы углубим свои знания о строках. Теперь мы сможем не только считывать строку, но и работать с ней, в том числе делать посимвольный перебор.*

**1. Строка как коллекция**

На прошлом занятии мы познакомились с коллекцией, которая называется **множество**. Вспомним, что основная особенность коллекций — возможность хранить несколько значений под одним именем. Можно сказать, что коллекция является **контейнером** для этих значений.

Но ещё до изучения множеств мы уже знали тип данных, который ведёт себя подобно коллекции. Этот тип данных — строка. Действительно, ведь строка фактически является последовательностью символов. В некоторых языках программирования есть специальный тип данных **char**, позволяющий хранить один символ. В Python такого типа данных нет, поэтому можно сказать, что строка — это **последовательность односимвольных строк**.

**2. Что мы знаем о строках**

Давайте вспомним, что мы уже знаем о работе со строками в языке Python. Мы умеем создавать строки четырьмя способами: задавать напрямую, считывать с клавиатуры функцией input(), преобразовывать число в строку функцией str и склеивать из двух других строк операцией +. Кроме того, мы умеем узнавать длину строки, используя функцию len, и проверять, является ли одна строка частью другой, используя операцию in:

fixed\_word = 'опять'

**print**(fixed\_word)

word = input()

**print**(word)

number = 25

number\_string = str(number)

**print**(number\_string)

word\_plus\_number = fixed\_word + number\_string

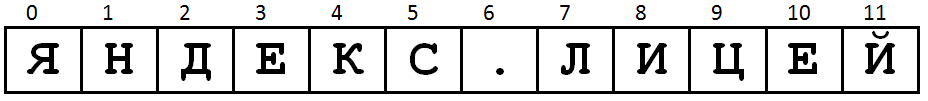
**print**(word\_plus\_number)

**print**(len(word\_plus\_number))

**print**('оп' **in** word\_plus\_number)

**3. Индексация в строках**

В отличие от множеств, в строках важен порядок элементов (символов). Действительно, если множества {1, 2, 3} и {3, 2, 1} — это одинаковые множества, то строки «МИР» и «РИМ» — это две совершенно разные строки. Наличие порядка даёт нам возможность пронумеровать символы. Нумерация символов начинается с 0:



**Важно**

По индексу можно получить соответствующий ему символ строки. Для этого нужно после самой строки написать в квадратных скобках индекс символа.

word = 'привет'

initial\_letter = word[0]

**print**(initial\_letter) *# сделает то же, что print('п')*

other\_letter = word[3]

**print**(other\_letter) *# сделает то же, что print('в')*

Естественно, в этом примере word с тем же успехом можно было считать с клавиатуры через input(). Тогда мы не могли бы заранее сказать, чему равны переменные initial\_letter и other\_letter.

А что будет, если попытаться получить букву, номер которой слишком велик? В этом случае Python выдаст ошибку:

word = 'привет'

**print**(word[6]) *# builtins.IndexError: string index out of range*

Конечно, номер в квадратных скобках — не обязательно фиксированное число, которое прописано в самой программе. Его тоже можно считать с клавиатуры или получить в результате арифметического действия.

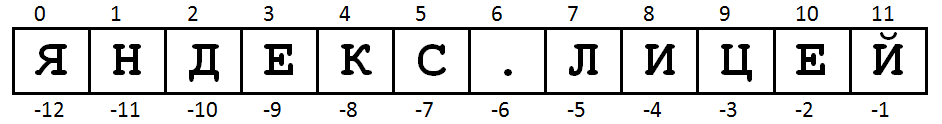
word = 'привет'

number\_of\_letter = int(input()) *# предположим, пользователь ввёл 3*

**print**(word[number\_of\_letter]) *# тогда будет выведена буква 'в'*

**Важно**

Кроме «прямой» индексации (начинающейся с 0), в Python разрешены отрицательные индексы: word[-1] означает последний символ строки word, word[-2] — предпоследний, и так далее.



А можно ли, используя индексацию, изменить какой-либо символ строки? Давайте проверим:

word = 'карова' *# Написали слово с ошибкой*

word[1] = 'о' *# Пробуем исправить, но:   
# TypeError: 'str' object does not support item assignment*

**Важно**

Интерпретатор Python выдаёт ошибку — значит, изменить отдельный символ строки невозможно, т.е. строка относится к неизменяемым типам данных в Python.

**4. Перебор элементов строки**

В предыдущем уроке мы узнали, что цикл for можно использовать для перебора элементов множества. Таким же образом можно использовать цикл for, чтобы перебрать все буквы в слове:

text = 'hello, my dear friends!'

vowels = 0

**for** letter **in** text:

**if** letter **in** {'a', 'e', 'i', 'o', 'u', 'y'}:

vowels += 1

**print**(vowels)

Но, так как символы в строке пронумерованы, у нас есть ещё один способ перебрать все элементы в строке — перебрать все индексы, используя уже знакомую нам конструкцию for i in range(...).

text = 'hello, my dear friends!'

vowels = 0

**for** i **in** range(len(text)):

**if** text[i] **in** 'aeiouy':

vowels += 1

**print**(vowels)

**5. Хранение текстов в памяти компьютера**

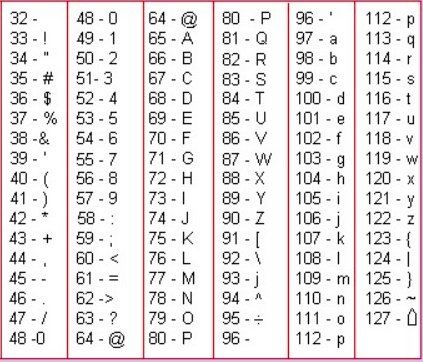
Давайте немного поговорим о том, как строки хранятся в памяти компьютера.

**Важно**

Поскольку компьютер «умеет» хранить только двоичные числа, для записи нечисловой информации (текстов, изображений, видео, документов) прибегают к **кодированию**.

Самый простой случай кодирования — сопоставление кодов текстовым символам.

Один самых распространенных форматов такого кодирования — таблица ASCII (American standard code for information interchange).



Изначально в этой таблице каждому символу был поставлен в соответствие 7-битный код, что позволяло идентифицировать 128 различных символов. В таблице вы не видите символы с кодами, меньшими 32, так как они являются служебными и не предназначены для непосредственного вывода на экран (пробел, перевод строки, табуляция и т.д.).

Этого хватало на латинские буквы обоих регистров, знаки препинания и спецсимволы — например перевод строки или разрыв страницы. Позже код расширили до 1 байта, что позволяло хранить уже 256 различных значений: в таблицу помещались буквы второго алфавита (например, кириллица) и дополнительные графические элементы (псевдографика).

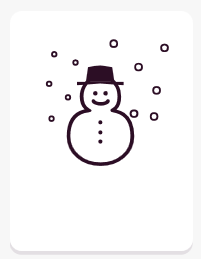
В некоторых относительно низкоуровневых языках (например в С) можно в любой момент перейти от представления строки в памяти к последовательности байтов, начинающейся по какому-либо адресу.

Сейчас однобайтные кодировки отошли на второй план, уступив место Юникоду.

**Юникод**

Юникод — это таблица, которая содержит соответствия между числом и каким-либо знаком, причем количество знаков может быть любым. Это позволяет одновременно использовать любые символы любых алфавитов, а также дополнительные графические элементы. Кроме того, в Юникоде каждый символ помимо кода имеет некоторые свойства: например, буква это или цифра. Это позволяет более гибко работать с текстами.

В Юникод все время добавляются новые элементы, 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. Например, есть несколько снеговиков.



Этого вы можете увидеть, если наберете:

**print**('**\u**2603')

В консоли увидим снеговика:

>>>☃

Важно понять, что все строки в Python хранятся именно как последовательность юникод-символов.

**Важно**

Для того, чтобы узнать код некоторого символа, существует функция **ord** (*от order — порядок*).

ord('Б')

>>>1041

**Важно**

Зная код, всегда можно получить соответствующий ему символ. Для этого существует функция **chr** (*от character — символ*):

chr(1041)

>>>'Б'

Функции ord и chr часто работают в паре. Попробуйте, например, предположить, что будет выведено на экран в результате работы следующей программы:

**for** i **in** range(26):

**print**(chr(ord('A') + i))

[Справка](https://yandex.ru/support/lyceum-students)

Исключительное право на учебную программу и все сопутствующие ей учебные материалы, доступные в рамках проекта «Яндекс.Лицей», принадлежат АНО ДПО «ШАД». Воспроизведение, копирование, распространение и иное использование программы и материалов допустимо только с предварительного письменного согласия АНО ДПО «ШАД».

© 2018 – 2020  ООО «[Яндекс](https://yandex.ru/)»

Чаты