

## [Инди-курс программирования на Python](#)

### 2.2 Строки: индексы и срезы

<https://stepik.org/lesson/295800/step/1?unit=277504>

**Строка** представляет собой упорядоченную коллекцию символов. Обратите внимание на слово «упорядоченную». То есть у каждого символа в строке есть свой порядковый номер и свое место.

### Индекс

Давайте разберем это на примере строки, состоящей из 11 символов `hello world`. У нас вначале идёт пять букв, далее идет один пробел и еще пять букв. Всего 2 слова и 11 символов. У каждого символа в этой строке есть свой порядковый номер. *Порядок нумерации в Python начинается с 0*, то есть самый левый символ `h` стоит на нулевом месте, затем буква `e` стоит на первом и так далее по порядку. То есть последняя буква у нас стоит на десятом месте. Порядковый номер также называется индексом и он нам нужен для того, чтобы получить значение, которое стоит на этом месте.

0	1	2	3	4	5	6	7	8	9	10
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
h	e	l	l	o		w	o	r	l	d

Создадим переменную `s` со значением `hello world`. Когда мы обращаемся к переменной `s` мы получаем все символы сразу, но если вдруг нам понадобится **получить только один символ, то тогда нам понадобится использовать индекс**. Например, чтобы получить `w` мы должны будем указать команду следующим образом: мы пишем строку, в которой мы хотим это получить, затем открываем квадратные скобочки, указываем номер индекса и получаем результат.

```
>>> s = 'hello world'
>>> s
'hello world'
>>> s[6]
'w'
```

То есть обращение у вас должно быть следующим образом: вы **пишете строку и в квадратных скобках вы указываете номер индекса**. Давайте попробуем получить букву `d`. Вы должны будете указать в квадратных скобках индекс номер 10:

```
>>> s[10]
'd'
```

То есть мы сейчас через переменную `s` обращаемся к индексу этой буквы, но вы также можете напрямую у самой строки обращаться к индексу. Рассмотрим вот такую строку `"abcd"` и мы можем взять, например, первый индекс - это будет буква `b`:

```
>>> 'abcd'[1]
'b'
```

Также вы **не можете обращаться к несуществующему индексу**. Если вы, например, хотите взять 14 символ, то получите вот такую ошибку, в которой указано, что индекс строки находится вне границ.

```
>>> s[14]
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    s[14]
IndexError: string index out of range
```

Давайте с вами в переменной **d** сохраним еще одну строку и запишем туда "privet". В этой строке такая же нумерация, то есть **первая буква стоит под нулевым индексом**, следующая буква стоит под первым и так далее. Последняя буква будет стоять под пятым индексом.

0	1	2	3	4	5
↑	↑	↑	↑	↑	↑
p	r	i	v	e	t

В этом примере мы с вами видим, что у всех строк можно одинаково получить первый символ - для этого нужно указать нулевой индекс, но последний символ у всех строк разный. То есть, например, в случае с переменной **s** - это был индекс под номером 10, а в случае с переменной **d** - это был индекс под номером 5.

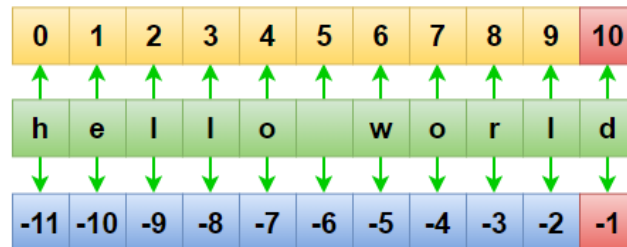
0	1	2	3	4	5
↑	↑	↑	↑	↑	↑
p	r	i	v	e	t

0	1	2	3	4	5	6	7	8	9	10
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
h	e	l	l	o		w	o	r	l	d

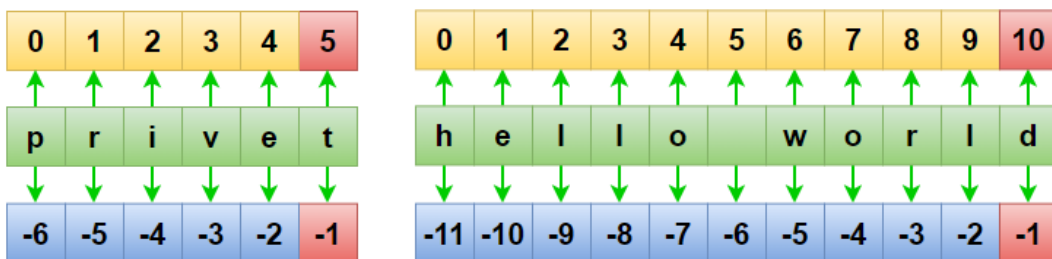
Номер последнего индекса меньше чем длина всей строки на 1. Следовательно, **последний индекс можно записать таким образом: в квадратных скобках вы находите длину вашей строки и вычитаете из неё один символ**.

```
>>> s[len(s) - 1]
'd'
>>>
>>> d[len(d) - 1]
't'
```

Таким образом вы можете получить последнюю букву, но на самом деле это очень сложный пример и чтобы его упростить в Python существует еще обратная нумерация. Она начинается с правого края и последняя буква стоит под индексом -1, затем будет стоять под -2 и так далее. Последняя будет стоять под индексом -11.



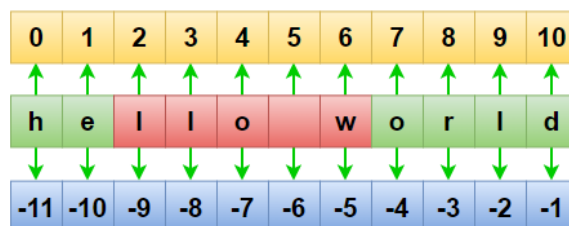
Отсюда мы можем сделать следующий вывод: чтобы **получить последний индекс вы можете указать просто -1**. И это будет работать в любой строке вне зависимости от ее длины.



```
>>> d[-1]
't'
>>> s[-1]
'd'
```

## Срез

Давайте обсудим что такое срез. Очень часто вам будет **необходимо взять в строке не один символ, а сразу несколько и для этого существуют срезы**. К примеру если в строке "hello world" вам потребуется взять вот эти вот символы (выделены красным):



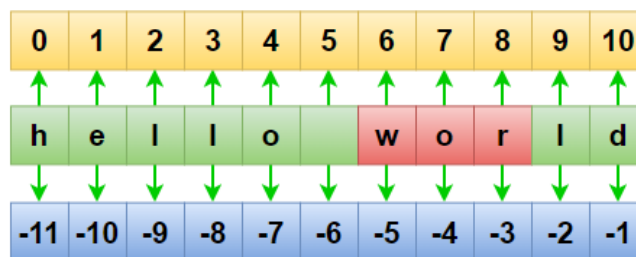
то вы должны будете написать следующим образом - вначале указываете у какой строки берем символы, открываете квадратные скобки, в них **указываете с какого индекса вам необходимо брать символы** и через двоеточие **указываете на каком индексе необходимо закончить**:

```
>>> s[1:6]
'ello '
```

Мы хотели взять по `w`, то есть у неё индекс 6 и мы видим с вами, что `w` не взялась и вам необходимо запомнить, что **при срезах у вас левое значение берется, а справа нет**. То есть для того, чтобы взялась `w` - вам необходимо указать следующий по порядку индекс - 7.

```
>>> s[1:7]
'ello w'
```

Давайте рассмотрим ещё один пример. Допустим, нам потребуется взять вот эти три буквы (выделено красным):



то мы должны начинать с 6 индекса и заканчивать 9, потому что 9 не возьмётся.

```
>>> s[6:9]
'wor'
```

В данном случае **при срезе мы передавали с вами два значения: начало интервала и конец**. Но вы можете не указывать один из них. Если мы с вами напишем вот таким образом `s[4:]`, то мы тем самым возьмем символы начиная с 4 индекса и по самый конец.

```
>>> s[4:]
'o world'
```

В данном случае берётся все начиная с 4 индекса и до самого конца. Также вы можете не указывать первый параметр, то есть оставить его пустым. Не забывайте поставить двоеточие. И не забывайте, что 4-ый индекс у нас братья не будет.

```
>>> s[:4]
'hell'
```

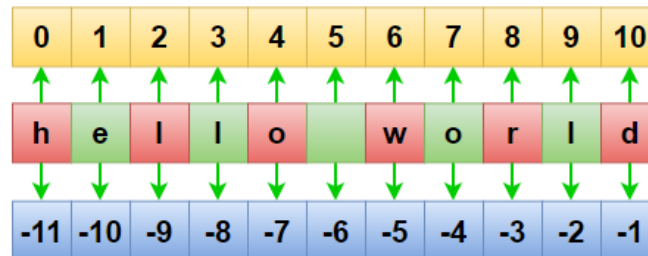
Не путайте когда вы пишете просто `s[4]`, то вы берете только один элемент под 4-ым индексом, а чтобы взять срез вы должны будете обязательно поставить двоеточие: `s[:4]` или `s[4:]`.

**Если вы напишете вот таким образом `s[:]`, то есть не укажете ни первый параметр, ни последний, то вы возьмете копию всей строки или всю строку целиком.**

```
>>> s[:]  
'hello world'
```

Также при срезах вы можете использовать третий параметр - это **шаг**.

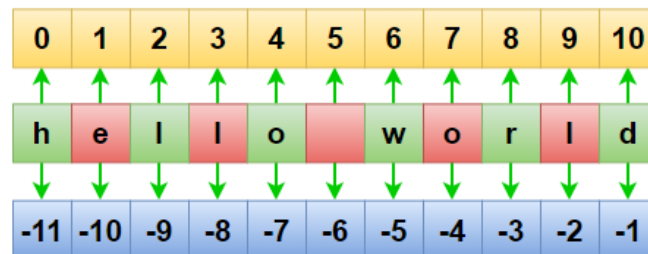
Например, у нас с вами всего 11 букв и мы хотим взять все буквы стоящие через одну:



Это можно сделать следующим образом: мы берем всю строку, то есть от начала до конца и третьим параметром передаем с каким шагом нужно брать следующую букву: `s[: :2]`. В данном случае шаг равен двум.

```
>>> s[: :2]  
'hlowrd'
```

Если вы хотите взять все остальные буквы, то в данном случае вы должны будете начинать с буквы `e`, то есть первого индекса и брать также через 1, то есть 3, 5, 7 и 9:



Для этого мы пишем следующим образом - в срезе указываем, что нужно взять строку с первого индекса и до конца, а также шаг 2:

```
>>> s[1: :2]  
'el ol'
```

Если вы укажете таким образом: `s[: : -1]`, то есть с самого начала и до конца и шагом -1, то перевернете строку:

```
>>> s[: : -1]  
'dlrow olleh'
```

Также вы можете писать таким образом `s[2:8:3]`, то есть указать начало 2, конец 8 и шаг 3 - мы получим с вами такой результат:

```
>>> s[2:8:3]
'1 '
```

**Строка - это неизменяемый объект.** Например, вы можете спокойно получить первый символ эту букву `h`, но если вы попытаетесь записать в эту строку какой-либо другой символ, то есть присвоить по индексу, например, скажем букву `a`, то вы получите вот такую ошибку, в которой говорится, что строка не поддерживает изменения.

```
>>> s[0] = 'a'
Traceback (most recent call last):
  File "<pyshell#51>", line 1, in <module>
    s[0] = 'a'
TypeError: 'str' object does not support item assignment
```

То есть **в уже существующем объекте вы не можете исправлять какую-либо букву** и если вы хотите какую-либо букву изменить, то вы должны действовать по-другому - вы должны будете создать новую строку.

Если мы с вами в строке `s` хотим изменить букву `o` стоящую на четвертом месте на букву `a`, то мы можем срезом взять всё, что стоит до этой буквы `a`, то есть с самого начала и до 4 элемента, затем к этому срезу добавить нашу новую букву и затем взять срез, который находится после буквы `a`, то есть начиная с пятого элемента и до самого конца.

```
>>> s[:4]+'a'+s[5:]
'hella world'
>>> s
'hello world'
```

Таким образом, вы получите новую строку, но при этом мы видим, что наша прошлая строка не изменилась и чтобы сохранить изменение в переменной `s` вы должны будете в нее присвоить всё наше выражение и вот тогда мы получим новую строку:

```
>>> s[:4]+'a'+s[5:]
'hella world'
>>> s
'hello world'
>>> s = s[:4]+'a'+s[5:]
>>> s
'hella world'
```