Строки (str)

- ➔ Строки - специальный тип данных, который позволяет хранить в памяти упорядоченный набор символов.
- ➔ Строки в основном служат для того, чтобы программа могла взаимодействовать с человеком понятным ему способом.
- ➔ Строки используются не только для вывода сообщений на экран, но и для ввода значений от человека к программе.

Для обозначения строки можно использовать как одиночные кавычки, так и двойные. Главное использовать в конце строки тот же вид кавычек, который был использован в начале.

```
>>> 'spam eggs'  # одиночные кавычки
'spam eggs'
>>> 'doesn\'t'  # одиночные кавычки и апостроф внутри строки
"doesn't"
>>> "doesn't"  # ...или используйте двойные кавычки
"doesn't"
>>> '"Yes," he said.' #Одиночные кавычки по краям и двойные внутри.
'"Yes," he said.'
>>> "\"Yes,\" he said." #Двойные кавычки по краям и внутри
'"Yes," he said.'
>>> '"Isn\'t," she said.' #Одиночные по краям, двойные и одиночные внутри
'"Isn\'t," she said.'
```

Питон умеет склеивать строки:
>>> ‘Строка с ’+’продолжением’
‘Строка с продолжением’
>>> print(‘Строка с ’+’продолжением’)
Строка с продолжением

Строки, так же как и числа можно присваивать переменным.
>>> st = “Привет, “
Для ввода строки нужно использовать функцию **input()**
>>> ans = input(‘Введите своё имя:’) #ввод с клавиатуры
Введите своё имя:Вася
>>> print(st+ans+’!’) #склеивание строк
Привет, Вася!

Операции со строками

Встроенная функция **len** возвращает длину строки:

```
>>> s = 'supercalifragilisticexpialidocious'
>>> len(s)
34
```

Строки могут быть соединены, повторены:

```
>>> 3 * 'un' + 'ium'
'ununununium'
```

Два или более строковых литерала расположенных друг за другом, автоматически соединяются в одну строку:

```
>>> 'Py' 'thon'
'Python'
```

Если использовать тройную кавычку, то строку можно записать на нескольких строчках:

```
print("""\
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
""")
```

Можно получать доступ к символам строки по их индексам. Первый индекс — ноль.

```
>>> word = 'Python'
>>> word[0]  # символ на позиции 0
'P'
>>> word[5]  # символ на позиции 5
'n'
>>> word[-1]  # последний символ
'n'
>>> word[-2]  # предпоследний символ
'o'
>>> word[-6]
'P'
```

Обратите внимание, что т.к.-0 равен 0, поэтому отрицательные индексы начинаются с -1

В дополнение к доступу по индексу, можно получать части (куски) строки.

```
>>> word[0:2]  # символы с позиции 0 (включая) по 2-ю (исключая)
'Py'
>>> word[2:5]  # символы с позиции 2 (включая) по 5 (исключая)
'tho'
```

Обратите внимание, что т. к. стартовый индекс всегда включается, а финишный никогда, то такая конструкция:
s[:i] + s[i:] всегда эквивалентна целой строке.

```
>>> word[:2] + word[2:]
'Python'
>>> word[:4] + word[4:]
'Python'
```

Можно использовать индексы по умолчанию:

```
>>> word[ :2 ]   # с начала до позиции 2 не включая
'Py'
>>> word[ 4: ]   # символы с четвертой позиции, включая до конца строки
'on'
>>> word[ -2: ]  # символы с предпоследней позиции до конца строки
'on'
```

P	y	t	h	o	n
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

Строки **неизменяемые**. Попытка изменить какой либо символ приведет к ошибке.

```
>>> word[:1] = 'J' # Так нельзя, будет ошибка
```

Если нужно изменить строку, то следует создать новую:

```
>>> 'J' + word[1:]
'Jython'
>>> word[:2] + 'py'
'Pypy'
```

Шаг нарезки строки (slicing step)

Кроме указания начала и конца интервала можно указать шаг, с которым будут будет сформирована строка.
Полный формат выглядит так **word[start : end : step]**
По умолчанию **step** равен единице и его можно не указывать. Вот две эквивалентных записи:
word[: -2 : 1] # вывод Pyth
word[: -2] #вывод Pyth

Вывод символов через один:
word[: : 2] #Вывод Pto

Вывод символов через один, начиная со второго символа:
word[1: : 2] # Вывод yhn

Отрицательный шаг

Шаг может быть и отрицательным, в этом случае строка выводится задом наперед.
word[:: -1] #Вывод nohtyP

Отрицательный шаг может быть и другим:
word[:: -2] #Ввод nhy

Нужно принимать во внимание, что при таком развертывании строки для получения срезов строки лучше сначала “развернуть” строку, а затем получит нужную ее часть.

word[::-1][:3] #Вывод noh
Так мы получаем три символа с конца строки. То есть сначала создается строка **nohtyP,** затем мы обычным способом получаем строку состоящую из первых трех символов.