Строки

Если мы что-то хотим привести к строке, то оборачиваем это в **str,** например:

```
p = str(input())
```

Но, **input()** оборачивать не нужно, тк это по дефолту строка

Строку нельзя изменять.
Если вы хотите обратиться к конкретному символу и изменить его, то у вас это не получится.
Но, мы можем вывести конкретный символ из строки, используя индексацию (индексы идут с нуля):

```
tmp = 'abcdef'
print(tmp[0])  #индекс указывается в квадратных скобках
```

Строки можно складывать.

```
s1 = 'fhdjdksf'
s2 = 'fjgdfvnd'
s3 = s1 + s2
print(s3)
```

Можно посчитать длину строки.

```
s1 = 'vhjbvjfiwjfpwifp'
print(len(s1))
```

Можно вывести срез строки.

```
s1 = 'fjhdfiovjefn'
print(s1[0:8:2])  #1 - начальный индекс, 2 - конечный индекс, 3 - шаг
```

Данная команда выведет **fhfo**.

Срезы в обратном порядке.

```
s1 = 'fjgnrjvnfjv'
print(s1[5:0:-2])
```

Данная команда выведет **jnj**.

Если выведем [5:-1:-2], то получим пустую строчку. Нужно будет убрать второй параметр (то есть, он не равен ничему), это будет по дефолту означать, что это до начала нашей строки (или, если поставим положительный шаг, то это будет означать до конца нашей строки)

Можно пропускать и другие параметры, например:

[:5] - выводим срез от начала до 5 индекса.
[3:] - от 3 до конца.

* по дефолту шаг будет всегда 1.

Разделение на подстроки.

```
tmp = "3 + 8 + 9 + 0"  #нет разницы между двойными и одинарными ковычками
t = tmp.split(" + ")
print(t)
```

Выведет ['3', '8', '9', '0'].

Соединение строк.

```
tmp = ('3' + '8' + '9' + '0')
print(' + '.join(tmp))
```

Форматирование.

Форматирование строк — это оформление строк с помощью методов форматирования, предложенных конкретным языком программирования. В Python таких несколько, но речь пойдет только об f-строках.

Каждая инструкция f-строки состоит из двух частей: символа `f` и строки, которую нужно отформатировать. Строка должна быть в одинарных, двойных или тройных кавычках.

```
f"string"
```

Переменные в фигурных {} скобках отображаются в выводе как обычно в print. Стоит посмотреть на примере.

```
# объявление переменных
name = input()
job = input()

# заключите переменную в {}, чтобы отобразить ее значение в выводе
print(f"{name} is a {job}.")

#также, можно выполнять арифметические операции
print(f"{2 * 2}")  #4
```

Как сделать так, чтобы все буквы стали заглавными.

```
s1 = "jvfjcqpdxwefwv"
print(s1.upper())  #JVFJCQPDXWEFWV
```

И строчными.

```
s1 = "GHRBEUYKJMIUKOI"
print(s1.lower())  #ghrbeuykjmiukoi
```

Символы.

В компьютере символы хранятся в виде таблицы, где каждый знак имеет свой номер.

Чтобы получить номер какого-либо символа, нужно ввести следующую команду.

```
print(ord('A'))  #у заглавных и строчных букв номера отличаются
```

Если мы хотим по номеру символа вывести сам символ, то пользуемся следующей командой:

```
print(chr(100))
```

И еще команды, выполняемые со строками:

- **isalpha()**: возвращает True, если строка состоит только из алфавитных символов
- **islower()**: возвращает True, если строка состоит только из символов в нижнем регистре
- **isupper()**: возвращает True, если все символы строки в верхнем регистре
- **isdigit()**: возвращает True, если все символы строки - цифры
- **isnumeric()**: возвращает True, если строка представляет собой число
- **startswith(str)**: возвращает True, если строка начинается с подстроки str
- **endswith(str)**: возвращает True, если строка заканчивается на подстроку str
- **lower()**: переводит строку в нижний регистр
- **upper()**: переводит строку в вехний регистр
- **title()**: начальные символы всех слов в строке переводятся в верхний регистр
- **capitalize()**: переводит в верхний регистр первую букву только самого первого слова строки
- **lstrip()**: удаляет начальные пробелы из строки
- **rstrip()**: удаляет конечные пробелы из строки
- **strip()**: удаляет начальные и конечные пробелы из строки
- **ljust(width)**: если длина строки меньше параметра width, то справа от строки добавляются пробелы, чтобы дополнить значение width, а сама строка выравнивается по левому краю
- **rjust(width)**: если длина строки меньше параметра width, то слева от строки добавляются пробелы, чтобы дополнить значение width, а сама строка выравнивается по правому краю
- **center(width)**: если длина строки меньше параметра width, то слева и справа от строки равномерно добавляются пробелы, чтобы дополнить значение width, а сама строка выравнивается по центру
- **find(str[, start [, end])**: возвращает индекс подстроки в строке. Если подстрока не найдена, возвращается число -1
- **replace(old, new[, num])**: заменяет в строке одну подстроку на другую