


# ПАРАМЕТРЫ STOP И END

---

 Модуль 1. Занятие 5.



# Тема **и цель**

---

## **Тема:**

Работа с параметрами `sep` и `end`.  
(в программе опечатка `stop`)

## **Цель занятия:**

Изучение способов работы  
с командой `print`.



# Глоссарий

---

- 1. Команда (в программировании)** — это указание компьютерной программе действовать как некий интерпретатор для решения задачи.



# Подтемы

---

1. Что такое **конкатенация строк**?
2. Как узнать **длину строки**?
3. Что можно проверить с помощью **print()** и **type()**.
4. Что такое **sep**?
5. Что такое **float**?



# Параметр `sep`

```
main.py
1 print('k', 'l', 'm')
2 print('n', 'e', 'z')
```

Output:

```
k l m
n e z
```

The screenshot shows a code editor window titled 'main.py' with two lines of Python code. The first line is `print('k', 'l', 'm')` and the second line is `print('n', 'e', 'z')`. Below the code editor, there is a toolbar with three icons: a downward arrow, a double-headed arrow, and a clipboard icon. The output of the code is displayed in a terminal window below the toolbar, showing two lines of text: `k l m` and `n e z`.

Рисунок 1



# Параметр `sep`

```
main.py
1 print('k', 'l', 'm', sep='*')
2 print('n', 'e', 'z', sep='*')
```

▼ ↗ 📄

```
k*l*m
n*e*z
```

Рисунок 2



# Параметр `sep`

---

1. На рисунке 1 в качестве строки разделителя между аргументами команды `print()` установлена строка `sep=''`.
2. На рисунке 2 в качестве строки разделителя между аргументами команды `print()` установлена строка `sep='*'`.
3. Таким образом, необязательный параметр `sep` команды `print()` позволяет установить строку, с помощью которой будут разделены аргументы на консоль.



# Параметр `end`

---

Если перевод строки делать не нужно или требуется указать специальное окончание, то следует явно указать значение для параметра `end`.

Рассмотрим следующий код и посмотрим на вывод:





# ЛИСТИНГ КОДА

```
main.py
1 print('k', 'l', 'm', end='@')
2 print('n', 'e', 'z', end='@@@')
3
k l m@n e z@@@
```

Рисунок 3



# ЛИСТИНГ кода

---

На рисунке 3 в первой строке вставлена строка @ вместо перевода строки. Аналогично, по завершении вывода второй строки вставлена строка @@.

Параметры sep и end можно использовать вместе.

Рассмотрим следующий код и обратим внимание на вывод:



# ЛИСТИНГ КОДА

```
main.py
1 print('s', 'y', 'n', sep='***', end='///')
2 print('e', 'r', 'g', sep='--', end=':~)')
3 print('y', 's', 't', sep='%', end='%%~)')
4 print('a', 'r', 't', sep='+', end='=')
5 print('p', 'y', 't', sep='$', end='finish|')
6
input
s***y***n///e--r--g:~)y%st%%~)a+r+t=p$y$tfinish
```

Рисунок 4



# Важно!

---

1. Вызов команды `print()` с пустыми скобками ставит перевод строки.
2. Последовательность символов `\n` называется управляющей последовательностью и задает перевод строки.
3. Значения по умолчанию у параметров `sep` и `end` следующие:

```
main.py
1  sep=' '      # пробел
2  end='\n'     # перевод строки
3
```



# Важно!

---

4. Чтобы убрать все дополнительные выводимые символы, можно вызывать команду `print()` так:

```
main.py
1 print('k', 'l', 'm', sep='', end='')
2
klm
```



# Важно!

---

## 5. Программный код

```
main.py
1 print('synergy')
2
▼ ↗ 📄
synergy
```

равнозначен коду

```
main.py
1 print('synergy', end='\n')
2
▼ ↗ 📄
synergy
```



# Важно!

---

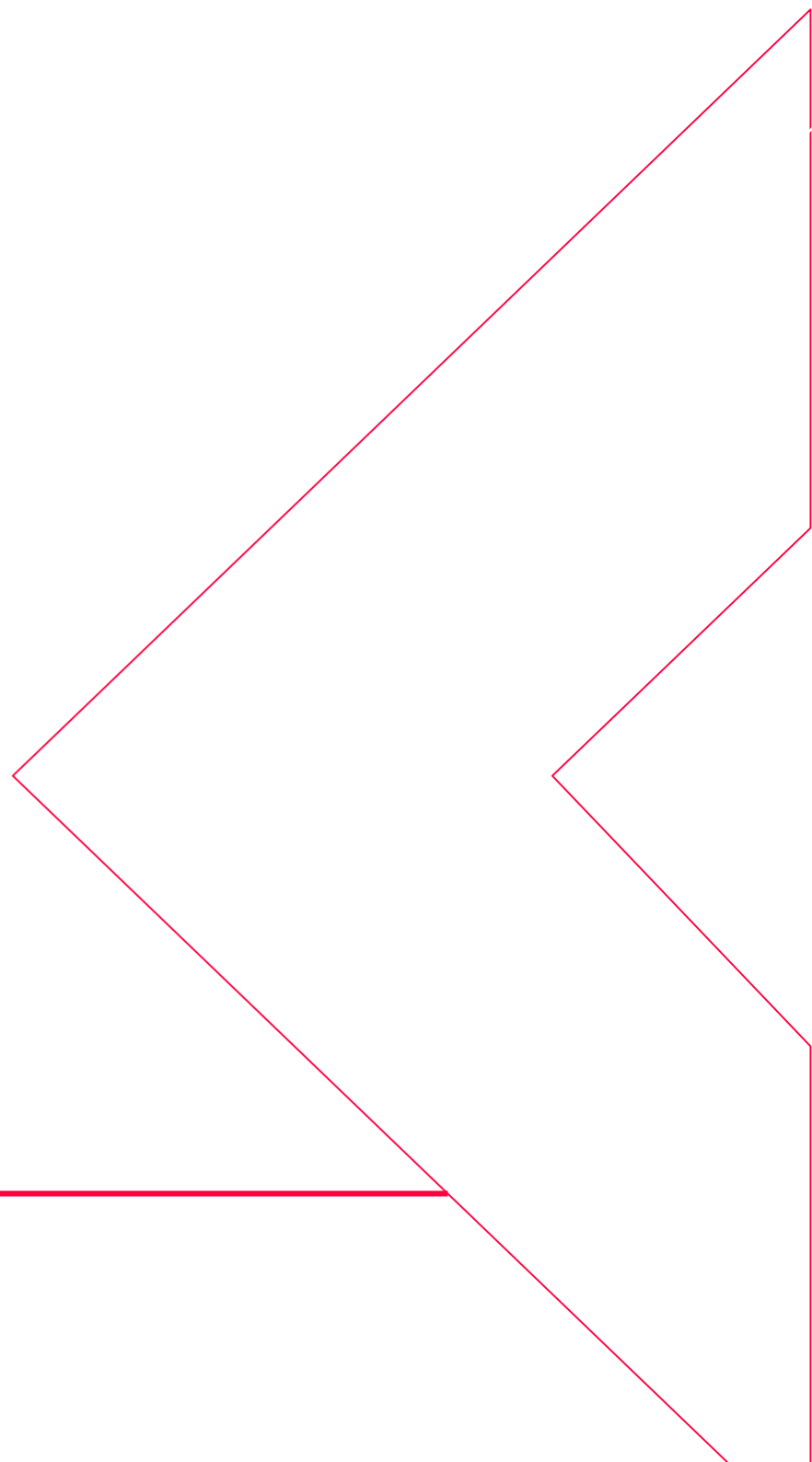
6. Если после вывода данных нужно более одного перевода строки, то необходимо использовать следующий код:

```
main.py
1 print('synergy', end='\n\n\n')
2
▼ ↗ 📄
synergy
```



# Практические задачи

---







# Задача 1

---

Напишите программу, которая выводит прямоугольник, состоящий из решеток (#).

*Примечание: высота и ширина прямоугольника произвольная.*



# Решение

Решение задачи 1 самым простым способом.  
Допускается решение различными способами.

```
main.py
1 print('#####')
2 print('#          #')
3 print('#          #')
4 print('#          #')
5 print('#####')
6
in
#####
#          #
#          #
#          #
#####
```



# Задача 2

---

Выведите на консоль строку с использованием функции `end`.



# Решение

---

Для решения данной задачи понадобится следующий код:

```
main.py
1 print('ветер', 'по', 'морю', 'гуляет', end=" ")
2 print('и', 'кораблик', 'подгоняет')
```

input

```
ветер по морю гуляет и кораблик подгоняет
```



# Задача 3

---

Выведите на консоль строку с использованием `sep='\n'`.



# Решение

---

Напишем код для решения данной практической задачи:

```
main.py
1 print('ветер', 'по', 'морю', 'гуляет', sep='\n')
2
```

input

ветер  
по  
морю  
гуляет



# Задача 4

---

Выведите на консоль строку с использованием `sep` и `end`.



# Решение

---

Для решения данной задачи понадобится следующий код:

```
main.py
1 print('ветер', 'по', 'морю', 'гуляет', sep='***', end="^^")
2 print('и', 'кораблик', 'подгоняет', sep='^^|', end="***")
```

input

```
ветер***по***морю***гуляет^^и^^кораблик^^подгоняет***
```





# Вопросы

---

1. Что такое **sep**?
2. Для чего применяется **end**?
3. Как вывести строку на консоль, разбив ее на две строки?
4. Для чего применяется функция **print**?
5. Как называется **последовательность символов \n**?