Тема урока: потоковый ввод stdin и вывод stdout

1. Потоковый ввод `stdin`
2. Потоковый вывод `stdout`

Аннотация. Урок посвящен потоковому вводу и выводу.

Потоковый ввод

В Python существует один очень полезный встроенный объект, который называется поток ввода (`sys.stdin`).

Поток ввода (`sys.stdin`) — это специальный объект в программе, куда попадает весь текст, который ввёл пользователь. Потоком его называют потому, что данные хранятся в нем до тех пор, пока программа их не прочитала. Таким образом, данные поступают в программу и временно сохраняются в потоке ввода (`sys.stdin`), а программа может забрать их оттуда, например, с помощью встроенной функции `input()` . В момент прочтения, данные пропадают из потока ввода, так как он хранит их до тех пор, пока они не будут прочитаны.

Поток ввода (`sys.stdin`) — является **итератором**, который невозможно перезапустить. Как и любой итератор, он может двигаться только вперёд. Как только данные прочитаны, они удаляются из потока ввода безвозвратно.

Элементы, которые выдает этот итератор — это строки, введённые пользователем. Если пользовательский ввод закончен, то итератор прекращает работу. Пока пользователь не ввёл последнюю строку, мы не знаем, сколько элементов в итераторе.

Мы с вами уже работали с итераторами, когда изучали встроенные функции `map(), filter(), zip()` . Итераторы будут подробно рассмотрены в этом курсе, но чуть позже. Пока достаточно помнить, что итератор — это специальный объект, элементы которого можно перебирать циклом `for` .

Чтобы работать с потоком ввода (`sys.stdin`), необходимо подключить модуль `sys` стандартно командой `import sys` .

Напишем программу, которая дублирует каждую введённую пользователем строку.

```
import sys

for line in sys.stdin:
    print(line.strip('\n'))
```

Строковый метод `strip('\n')` отрезает от строки `line` символ перевода строки, поскольку функция `print` сама переводит строку.

Пока есть данные в потоке ввода `sys.stdin` (то есть пока пользователь их вводит) программа будет записывать вводимые строки в переменную `line`, убирать символы перевода строки и выводить их на печать.

Если запустить такую программу, то она будет работать вечно. Чтобы показать, что ввод закончен, недостаточно нажать `Enter` — компьютер не знает, завершил ли пользователь работу или будет ещё что-то вводить (при этом `Enter` превратится в пустую строку). Для завершения ввода необходимо ввести `Ctrl + D` (если работаете в консоли Linux или IDE PyCharm), либо `Ctr + Z`, затем `Enter` (если работаете в консоли Windows).

Если вы работаете в IDE Wing, кликните правой кнопкой мыши и выберите Send EOF, затем нажмите `Enter`. Это запишет в поток ввода специальный символ EOF (end of file) который отмечает конец ввода.

Читаем входные данные в одну строку

С помощью потока ввода (`sys.stdin`) можно в одну строчку кода прочитать весь пользовательский ввод в список.

Реализуется это с помощью списочного выражения:

```
import sys

data = [line.strip() for line in sys.stdin]
```

или с помощью функции высшего порядка `map()`:

```
import sys

data = list(map(str.strip, sys.stdin))
```

Обратите внимание, что мы ничего не знаем о количестве введенных строк. Раньше приходилось в задачах сначала указывать количество строк, а уже затем сами строки.

Рассмотрим программный код, который дважды вызывает функцию `input()`:

```
name, surname = input(), input()
```

Если такой программе передать только одну строку, то выполнение программы завершится с ошибкой:

```
EOFError: EOF when reading a line.
```

поскольку второй вызов функции `input()` не смог ничего прочитать.

Таким образом если мы не знаем, в какой момент надо прекратить ввод, то воспользоваться функцией `input()` не удастся. В таких случаях остаётся только работать с потоковым вводом (`sys.stdin`).

Методы read() и readlines()

Как уже было сказано выше, мы можем обойти циклом `for` итератор `sys.stdin` . Кроме того, можно считать все строки из итератора (с сохранением символов перевода строки) в список с помощью метода `readlines()` :

```
import sys

data = sys.stdin.readlines()
```

Обратите внимание на то, что символ перехода на новую строку (`\n`) сохраняется в считанных строках.

Если разделять на строки нет необходимости, то считать многострочный текст из стандартного потока ввода в текстовую переменную можно с помощью метода `read()` :

```
import sys

data = sys.stdin.read()
```

Потоковый вывод

Аналогичным образом можно работать с потоковым выводом (`sys.stdout`). По умолчанию функция `print()` перенаправляет вывод данных именно в `sys.stdout` , хотя нам ничего не мешает самостоятельно писать в него.

Приведенный ниже код:

```
import sys

print('Hello')
sys.stdout.write('world!')
print('from')
sys.stdout.write('python\n')
print('Bye-bye')
```

выведет:

```
Hello
world!from
python
Bye-bye
```

Обратите внимание на то, что функция `print()` добавляет перевод на новую строку, а явная запись данных в `sys.stdout` с помощью метода `write()` нет. Чтобы добавить перевод на новую строку, мы используем стандартный символ `\n`.

Также нужно иметь в виду, что при использовании потока вывода `sys.stdout` нам нужно самостоятельно преобразовывать данные к строковому типу данных (функция `print()` это делает автоматически).

Приведенный ниже код:

```
import sys

sys.stdout.write(17)
```

приводит к возникновению ошибки.

Исправленная версия кода:

```
import sys

sys.stdout.write(str(17))     # преобразуем данные в строку
```

Примечания

Примечание 1. По умолчанию функция `input()` читает данные из потока ввода `sys.stdin`, а функция `print()` печатает данные в поток вывода `sys.stdout`.

Примечание 2. Функция `print()` — это удобная обертка (wrapper) вокруг метода `sys.stdout.write()`. Функцию `input()` часто можно рассматривать как обертку (wrapper) вокруг `sys.stdin.readline()`.

Примечание 3. Объекты `sys.stdin` и `sys.stdout` являются файловыми объектами, предоставляемыми ОС. Им доступны все соответствующие методы (`read(), readline(), readlines(), write(), writelines()`). В общем случае, когда программа запускается в интерактивном сеансе, `stdin` является клавиатурным вводом, а `stdout` является выводом на экран, но оболочка может использоваться для перенаправления из обычных файлов или вывод на канал и ввода в другие программы.

Примечание 4. Приведенный ниже код:

```
import sys

temp = sys.stdout
sys.stdout = open('log.txt', 'w')
print('testing123')
print('another line')
sys.stdout.close()
sys.stdout = temp
print('back to normal')
```

приведет к созданию текстового файла `log.txt`, содержащего:

```
testing123
another line
```