Модуль 3 Занятие 3

Работа с файлами

Потоковый ввод

Чтение из файла

Контекстный менеджер

Запись в файл

Стандартный поток ввода

Стандартный поток ввода (sys.stdin) — это специальный объект, в который попадают все входные данные, передаваемые интерпретатору.

Стандартный поток ввода является итератором и хранит все входные данные до тех пор, пока они не будут считаны, а значит элементы ввода (строки) можно перебрать в цикле.

Чтобы использовать стандартный поток ввода stdin в программе, необходимо его импортировать из модуля sys:

```
from sys import stdin
```

Потоковый ввод

```
from sys import stdin

lines = []
for line in stdin:
    lines.append(line)
    # lines.append(line.strip('\n'))
print(lines)
```

Чтобы завершить ввод данных, необходимо воспользоваться сочетанием клавиш Ctrl + D (Ctrl + Z).

Пример ввода:

```
привет
как
дела?
^D
```

Пример вывода:

```
['привет\n', 'как\n', 'дела?\n']
```

Потоковый ввод

Можно считать все строки стандартного потока в список с помощью метода readlines().

```
from sys import stdin
lines = stdin.readlines()
print(lines)
```

Вывод:

```
['привет\n', 'как\n', 'дела\n']
```

Можно считать весь стандартный поток ввода в строку с помощью метода read().

```
from sys import stdin
lines = stdin.read()
print(lines)
```

Вывод:

```
['привет\nкак \nдела \n']
```

Важно! Ввод будет содержать символы «\n» в конце каждой введенной строки.

Пример

```
from sys import stdin


total = 0
for line in stdin:
    line = line.strip('\n')
    if line.isdigit():
        total += int(line)


print(total)
```

Считаем с клавиатуры произвольное количество строк и посчитаем сумму только тех строк, которые являются числами.

Работа с файлами

Для работы с файлами в Python существует функция open(), которая имеет следующие аргументы:

- file (обязательный аргумент) — строка, содержащая путь до файла, может быть относительным, например 'text.txt' или абсолютным 'C:/Documents/text.txt'.
- mode — режим доступа к файлу: 'r' — чтение (по умолчанию), 'w' — запись, 'a' — добавление и другие.
- encoding — строка, содержащая обозначение кодировки, например, 'utf-8'.

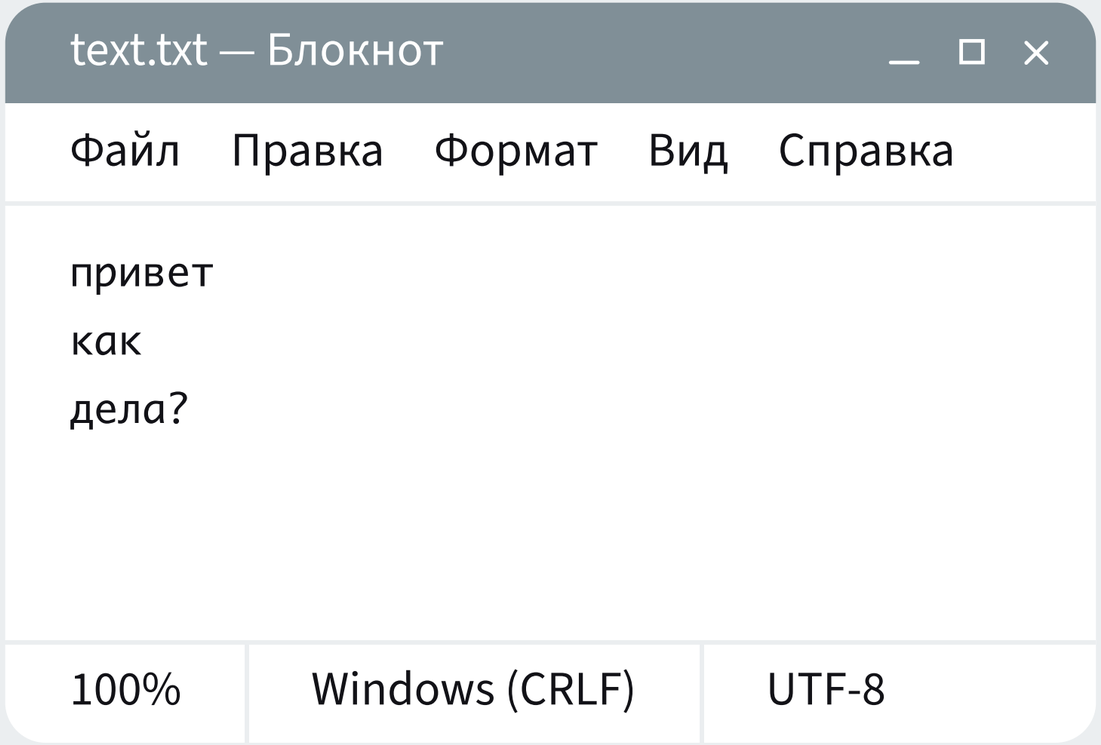
Функция open() возвращает файловый объект, являющийся итератором.

Чтение из файла

При работе с файлами необходимо выполнить 3 действия:

- открыть файл
- обработать файл
- закрыть файл

В папке с программой создайте текстовый файл text.txt с содержимым:




```
file = open(file='text.txt', mode='r', encoding='utf-8')
lines = []
for line in file:
    lines.append(line)
    # lines.append(line.strip('\n'))
print(lines)
file.close()
```

Вывод:

```
['привет\n', 'как \n', 'дела?\n']
```

Чтение из файла

Можно считать все строки файла в список с помощью метода readlines().

```
file = open('text.txt', encoding='utf-8')
lines = file.readlines()
print(lines)
file.close()
```

Вывод:

```
['привет\n', 'как\n', 'дела?\n']
```

Можно считать все строки файла в строку с помощью метода read().

```
file = open('text.txt', encoding='utf-8')
lines = file.read()
print([lines])
file.close()
```

Вывод:

```
['привет\nкак\nдела\n']
```

Важно! Ввод будет содержать символы «\n» в конце каждой введенной строки.

Закрытие файла

После работы с файлом его необходимо закрыть с помощью метода close().
Для этого есть несколько причин:

- Если в файл вносились изменения, то это позволит корректно его сохранить.
- Открытый файл может оказаться недоступен для других программ.
- Если файл уже не нужен, то стоит освободить память и не хранить ненужные данные.

Существует возможность закрывать файл автоматически после окончания работы с ним. Для этого используется менеджер контекста with.

Менеджер контекста with

Оператор with позволяет реализовать контекст исполнения определённого кода. Менеджер контекста — специальная конструкция управления к ресурсу, например, файлу. По выходу из контекста ресурс автоматически освобождается, даже если при выполнении блока возникло исключение. При использовании with нет необходимости закрывать файл вручную и, даже если во время выполнения кода внутри конструкции with возникнут ошибки, файл все равно будет закрыт.

```
with open('text.txt', encoding='utf-8') as file:
    lines = file.readlines()

# файл уже автоматически закрыт
print(lines)
```

При работе с файлами предпочтительно всегда использовать менеджер контекста.

Пример

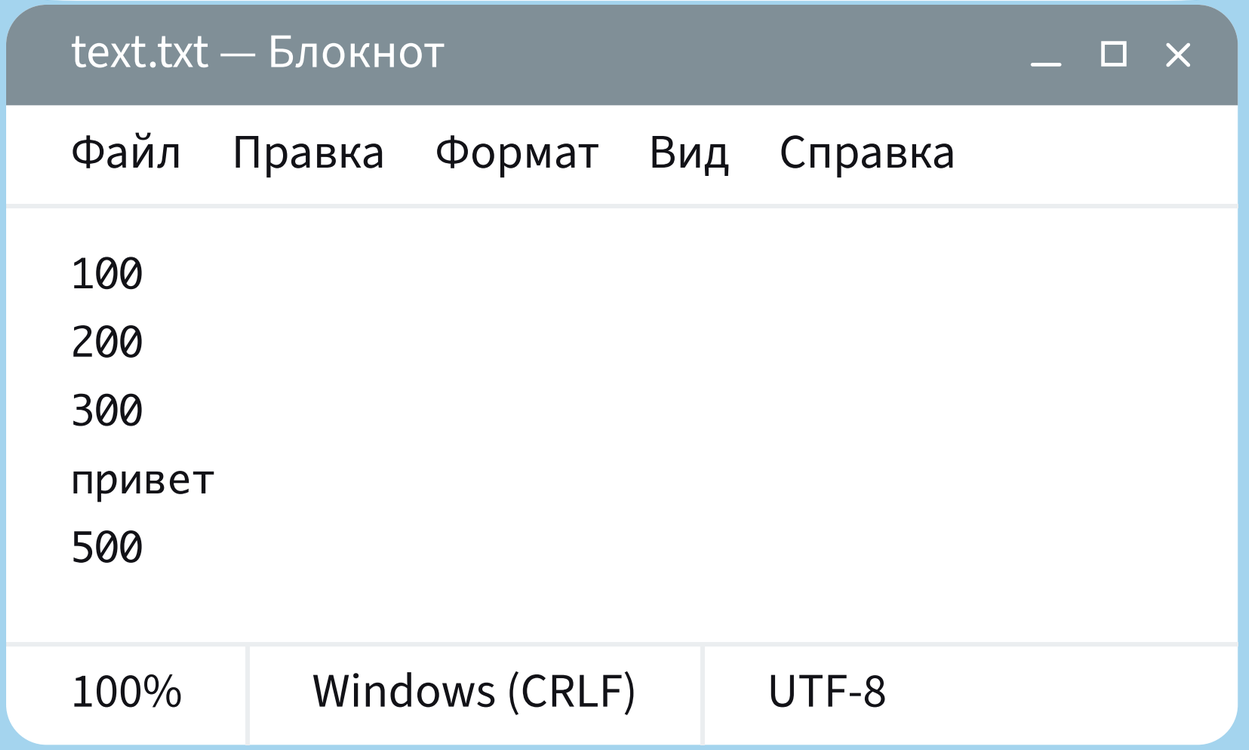
```
with open('text.txt', encoding='utf-8') as file:
    lines = file.readlines()

total = 0
for line in lines:
    line = line.strip('\n')
    if line.isdigit():
        total += int(line)

print(total)
```

Считываем из файла произвольное количество строк и посчитаем сумму только тех строк, которые являются числами.

В папке с программой создайте текстовый файл text.txt с произвольным содержимым:



Запись в файл

При записи в файл его необходимо открыть в режиме записи, выбрав режим доступа:

- 'w' — если файл уже существует, то его содержимое перезаписывается, если файла не существует, то он будет создан.
- 'a' — если файл уже существует, то его содержимое сохраняется, а новые данные добавляются в конец, если файла не существует, то он будет создан.

Для записи можно использовать методы write() или writelines():

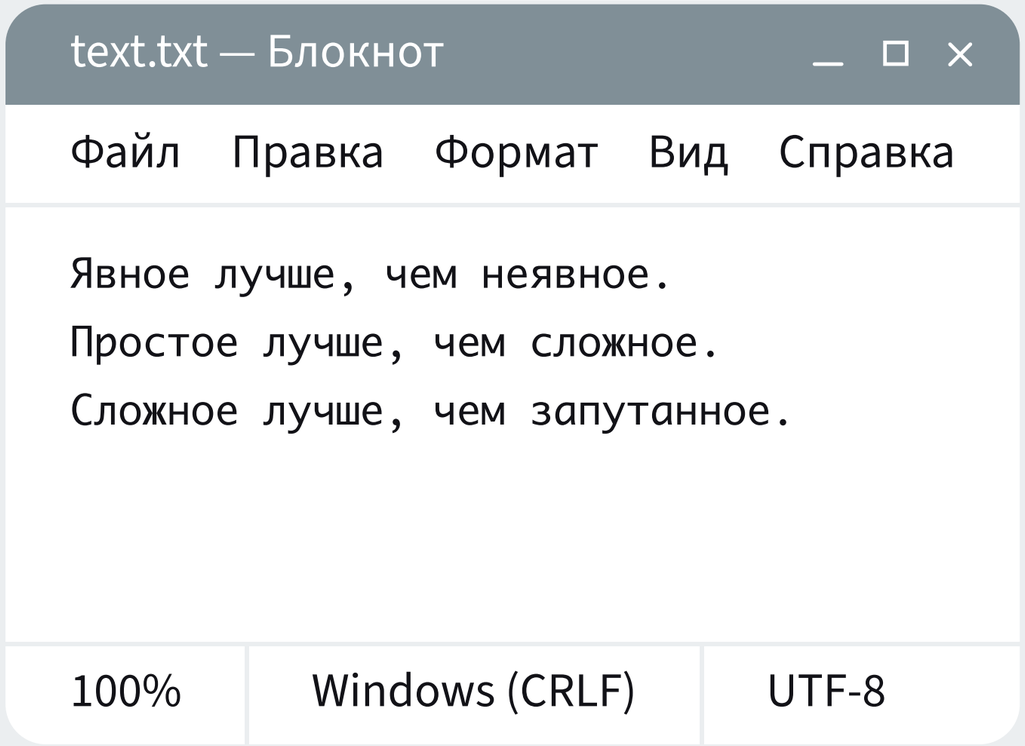
- метод write() — записывает переданную строку в файл.
- метод writelines() — записывает переданный список строк в файл.

Запись в файл

```
text1 = 'Явное лучше, чем неявное.\n'
text2 = 'Простое лучше, чем сложное.\n'
text3 = 'Сложное лучше, чем запутанное.\n'

with open('output.txt', 'w', encoding='utf-8') as file:
    file.write(text1)
    file.write(text2)
    file.write(text3)
```

В папке с программой будет создан файл output.txt с содержимым:

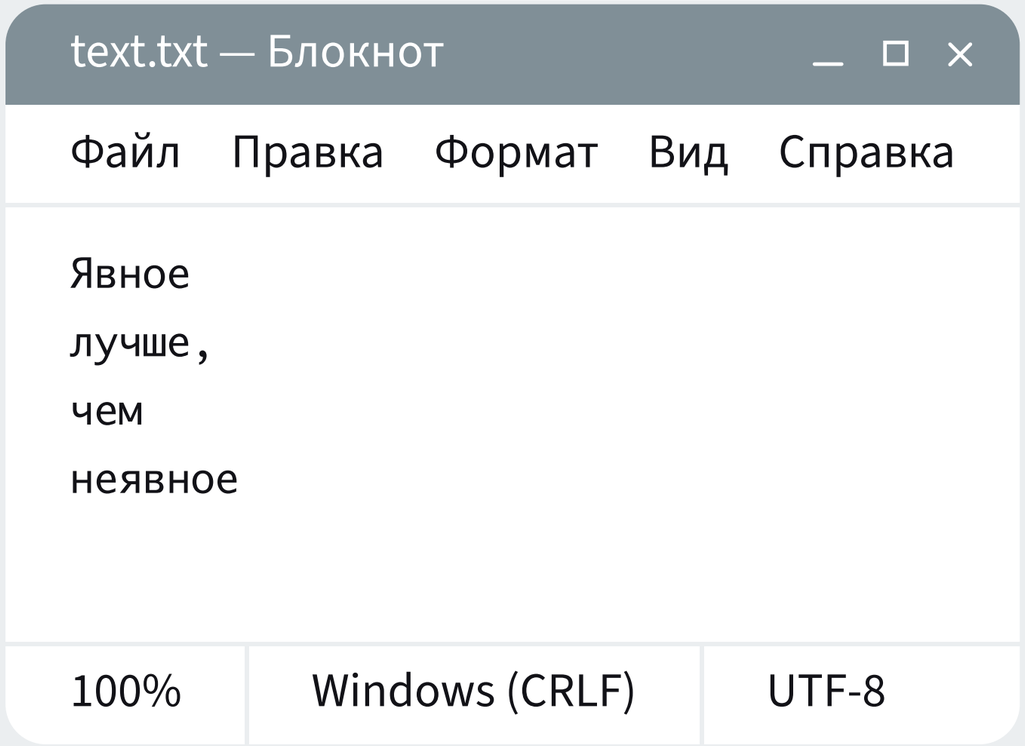


Запись в файл

```
text = ['Явное\n', 'лучше,\n', 'чем\n', 'неявное\n']


with open('output.txt', 'w', encoding='utf-8') as file:
    file.writelines(text)
```

В папке с программой будет создан файл output.txt с содержимым:

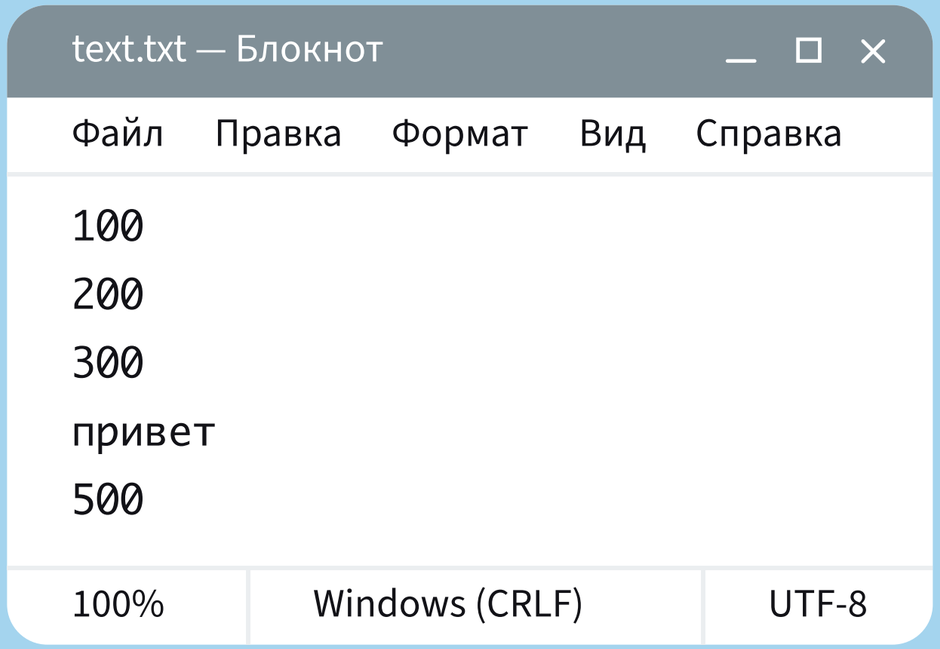


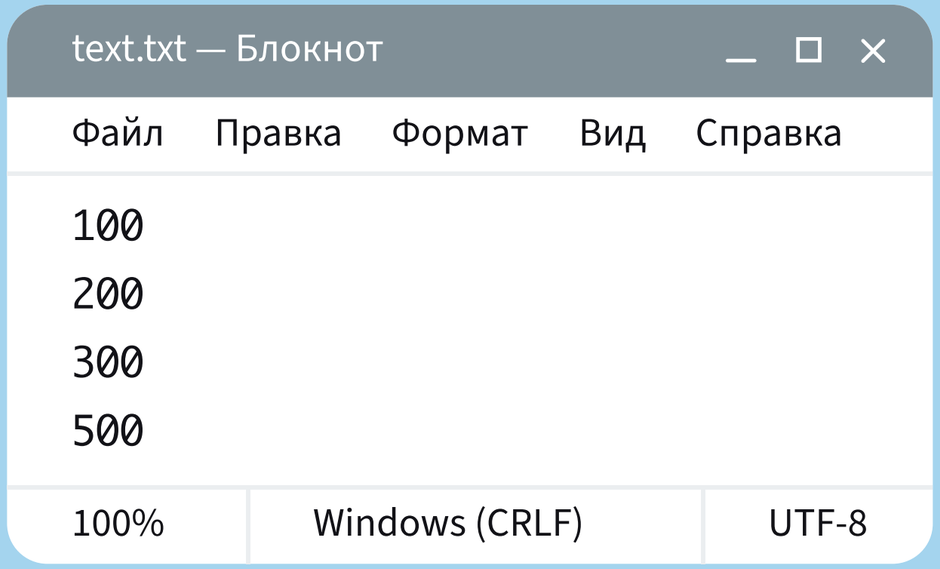
Пример

Считываем из файла произвольное количество строк и сохраним в другой файл только те строки, которые являются числами.

```
with open('input.txt', encoding='utf-8') as file_input:
    lines = file_input.readlines()

with open('output.txt', 'w', encoding='utf-8') as file_output:
    for line in lines:
        if line.strip('\n').isdigit():
            file_output.write(line)
```





Итоги

- Стандартный поток ввода (sys.stdin) позволяет считать с клавиатуры произвольное число строк.
- Для работы с файлами в Python существует функция open(), которая позволяет открыть файл для чтения или записи.
- При работе с файлами предпочтительно всегда использовать менеджер контекста with.
- Для чтения данных из файла его необходимо открыть в режиме чтения ('r' — по умолчанию) и использовать цикл по файловому объекту или методы read() или readlines().
- Для записи в файл его необходимо открыть в режиме записи ('w' или 'a') и использовать методы write() или writelines().