Ввод и вывод

Понятие файлов и потоков

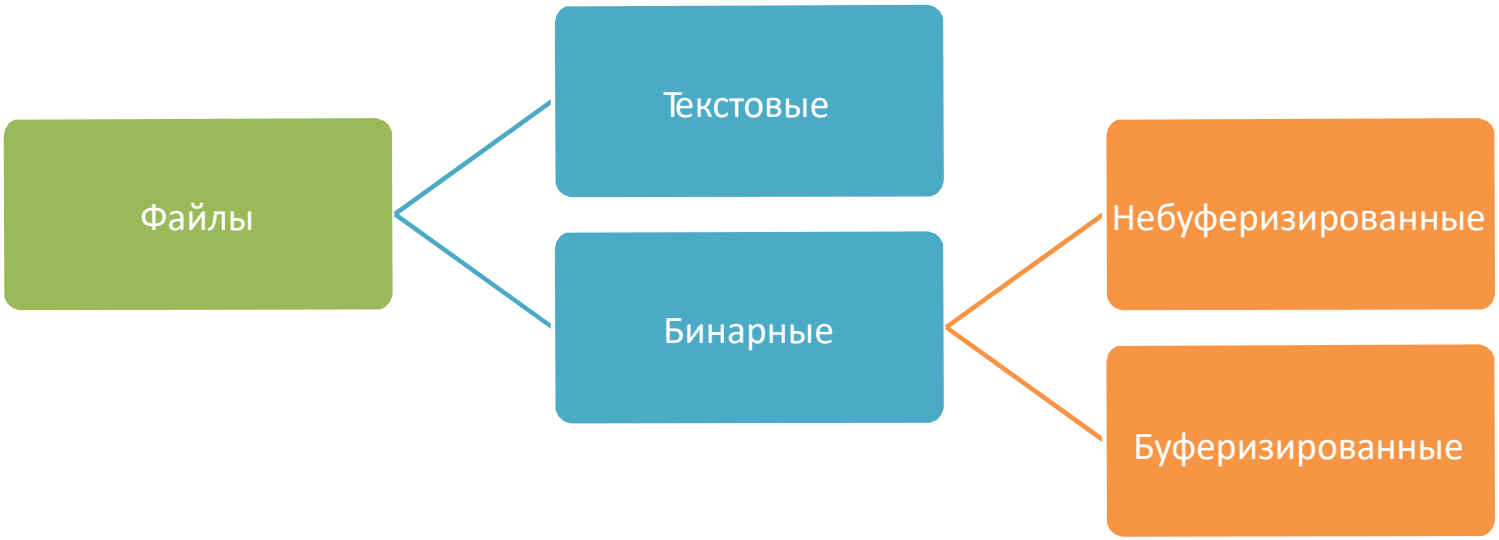
Файл – именованная область данных на носителе информации.

Файловый объект (поток) – объект, предоставляющий файл-ориентированный API (методы read(), write() и т.д.) для доступа к ресурсу. В зависимости от способа создания, файловый объект может предоставлять доступ к реальному файлу на диске или другому виду устройства хранения или передачи данных (стандартные потоки ввода/вывода, буферы в памяти, сокеты и т.д.).

Стандартные потоки:

- sys.stdin
- sys.stdout
- sys.stderr

Виды файлов в Python



Открытие файла

Python 2

```
open(file, mode='r', buffering=-1)
```

Python 3

```
open(file, mode='r', buffering=-1,
     encoding=None, errors=None,
     newline=None, closefd=True,
     opener=None)
```

Основные параметры:

- file – имя файла или файловый дескриптор;
- mode – режим открытия файла;
- encoding – кодировка файла;
- buffering – использовать ли буферизацию и если да, каким должен быть размер буфера

Чаще всего функция open() используется с двумя параметрами

Режимы открытия файлов

	Символ	Значение
первый символ	'r'	открыть для чтения (по умолчанию)
	'w'	открыть для записи, удалив содержимое файла
	'x'	открыть для исключительного создания (ошибка, если файл существует)
	'a'	открыть для записи, добавляя содержимое в конец файла, если он существует
второй или третий символ	'b'	бинарный режим
	't'	текстовый режим (по умолчанию)
	'+'	открыть для чтения и записи

Порядок символов ‘b’/‘t’ и ‘+’ не имеет значения

Закрытие файлов

После окончания работы с файлом следует обязательно его закрыть при помощи метода close(), особенно если он был открыт для записи. При использовании буферизированного вывода данные, которые записываются в файл, не попадают в него сразу, а записываются в буфер. Содержимое буфера записывается в файл при его заполнении или вызове методов flush() или close(). Кроме того, если файл открыт для записи, он будет заблокирован для открытия для записи другими процессами до момента закрытия. Все открытые файлы автоматически закрываются при удалении соответствующих файловых объектов из памяти сборщиком мусора интерпретатора Python и при завершении работы самого интерпретатора, однако следует держать файлы открытыми минимально требуемое время.

Оператор with

Файловые объекты являются менеджерами контекста.

Менеджер контекста (context manager) – это объект, который описывает определённый контекст, который устанавливается при выполнении оператора with. Менеджеры контекста используются для сохранения и восстановления глобального состояния, блокирования и разблокирования ресурсов, автоматического закрытия файлов и т.д.

Таким образом, при открытии файла таким образом:

```
with open(filename, mode) as file:
    …
    # perform actions on file
    …
```

он гарантированно будет закрыт, как только перестанет быть нужен.

Чтение файлов

Чтение одной строки текста	`file.readline()`
Чтение списка строк текста	`file.readlines()`
Построчное чтение	`for line in file:` `  pass`
Чтение заданного или максимально возможного количества данных (символов или байтов)	`file.read(100)` `file.read()`
Чтение данных бинарного файла в изменяемый массив	`file.readinto(arr)`

Запись в файлы

Запись строк текста	`file.writelines(lines)`
Запись данных (символов или байтов)	`file.write(data)`
Запись данных в текстовый файл при помощи функции или оператора print	`print(*args, file=text_file)  # Python 3` `print >>text_file, arg1, arg2, …  # Python 2`

Сериализация объектов

Сериализация – это процесс сохранения объектов в двоичном или строковом виде для хранения, передачи и восстановления. Обратный процесс называется десериализацией.

Модуль json

- Для сохранения разнообразных данных в текстовом виде можно использовать формат JSON (JavaScript Object Notation), который является стандартом де-факто в веб-приложениях и довольно популярен в других областях.
- Он поддерживается (на уровне языка, стандартной библиотеки языка или, по крайней мере, существующих сторонних библиотек) всеми современными языками программирования и знаком многим разработчикам, что делает его отличным выбором для сохранения данных для передачи между приложениями, написанными на разных языках.
- Модуль json позволяет сериализовать словари, списки, кортежи, строки, целые и вещественные числа, перечисления, булевские значения и None. Для поддержки других типов данных следует расширить классы JSONEncoder и JSONDecoder.