

Файлы python

Файл — это определенное количество информации, имеющее имя и хранящееся в долговременной памяти. Другими словами, это набор данных, которому дали имя и где-то сохранили.

В Питоне файлы делятся на две разновидности:

- Текстовые
- Бинарные

Текстовые файлы

Предполагается, что этот вид файлов содержит набор символов, который может прочитать человек. Как правило, такие файлы могут быть открыты любым простейшим текстовым редактором, к примеру, блокнотом в windows.

Бинарные файлы

Бинарные файлы содержат набор нулей и единиц. Таким образом можно хранить любую информацию: изображения, аудио, видео и даже текст.

Работа с файлами состоит из следующих шагов:

Файл надо открыть

Произвести необходимые операции (запись или чтение)

Закрыть файл

Открытие файла

Метод open

И так, если Вы хотите произвести какие-либо операции с файлом, сперва придётся его открыть. Для этой цели в языке Пайтон есть встроенная функция `open()`. Используя эту функцию, можно создать на основе любого языка файла объект Python.

```
file = open(name, mode)
```

Где,

- `name`- имя файла, который Вы открываете
- `mode`- режим открытия. Если не указать этот параметр, файл будет открыт в режиме «только чтение». Режимы открытия файла могут быть следующими:

| | |
|----------|--|
| r | Режим только для чтения. Указатель стоит в начале файла. |
| rb | Режим для чтения в двоичном формате. Указатель стоит в начале файла. |
| r+ | Режим для чтения и записи. Указатель стоит в начале файла. |
| rb+ | Режим для чтения и записи в двоичном формате. Указатель стоит в начале файла. |
| w | Режим только для записи. Указатель стоит в начале файла. Создает файл с именем имя_файла, если такового не существует. |
| wb | Режим для записи в двоичном формате. Указатель стоит в начале файла. Создает файл с именем имя_файла, если такового не существует. |
| w+ | Режим для чтения и записи. Указатель стоит в начале файла. Создает файл с именем имя_файла, если такового не существует. |
| wb+ | Режим для чтения и записи в двоичном формате. Указатель стоит в начале файла. Создает файл с именем имя_файла, если такового не существует. |
| a | Режим для добавления информации в файл. Указатель стоит в конце файла. Создает файл с именем имя_файла, если такового не существует. |
| ab | Режим для добавления в двоичном формате. Указатель стоит в конце файла. Создает файл с именем имя_файла, если такового не существует. |
| a+ | Режим для добавления и чтения. Указатель стоит в конце файла. Создает файл с именем имя_файла, если такового не существует. |
| ab+ | Режим для добавления и чтения в двоичном формате. Указатель стоит в конце файла. Создает файл с именем имя_файла, если такового не существует. |

Файловый объект имеет несколько атрибутов, предоставляющих информацию о файле:

| | |
|--------------------|---|
| file.closed | Выводит True если файл был закрыт. |
| file.mode | Выводит режим доступа, с которым был открыт файл. |
| file.name | Выводит имя файла. |
| file.softspace | Выводит False если при выводе содержимого файла следует отдельно добавлять пр |

Пример

Для начала необходимо сохранить в новый файл «test.txt» какой-то текст. Этот файл должен быть расположен в рабочей папке.

Применим следующий код для открытия данного файла:

```
file = open('test.txt', 'r') # открыть файл из рабочей папки в режиме
чтения, относительный путь
file_2 = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r') # открыть
файл из любого каталога
```

В переменных `file` и `file_2` хранятся ссылки на объекты с открытыми файлами. Теперь посмотрим содержимое файла и информацию о нём:

```
file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r')
print(*file)
print(file)
file.close()
# Вывод:
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

```
<_io.TextIOWrapper name='C:\\Users\\VVD\\Desktop\\test.txt' mode='r'
encoding='cp1251'>
```

Заккрытие файла

Метод `close`

Когда файл открывается, он начинает потреблять дополнительные ресурсы, поэтому, после выполнения всех операций, его необходимо закрыть.

Python сам закроет файл, если присвоить объект другому файлу.

Так же можно использовать метод `close()`.

```
file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r') # открыть файл
из любого каталога
print(*file)
file.close()
```

Инструкция `with`

В Питоне есть более удобный инструмент взаимодействия с файлами, чем те, которые использовались до этого — конструкция `with`. Кроме того, конструкция `with` гарантирует закрытие файла автоматически.

Конструкция `with` называется менеджер контекста.

Обратите внимание, насколько это интуитивно понятно. Оператор Python `with` всегда будет закрывать файл в конце, даже если программа завершилась ненормально даже в контексте или блоке. Эта функция безопасности рекомендуемый выбор для всех программистов. В таком случае инструкция `close` не нужна, потому что `with` автоматически закроет файл.

Пример использование менеджера контекста:

```
with open('C:\\Users\\VVD\\Desktop\\test.txt') as file:  
    # какие-то операции
```

Чтение и запись файлов в python

Python позволяет как читать, так и записывать в файлы информацию. Для этого при открытии применяются различные режимы.

Функция read

Применяя функцию read() Вы можете прочесть информацию из файл, который был открыт в режиме чтения (r):

```
file.read(size)
```

Где,

- file- объект файла
- size- количество символов, которые Вы хотите получить. Когда этот параметр не указан, файл читается от начала до конца.

Вот как это выглядит:

```
file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r')  
print(file.read(17))  
file.close()  
# Вывод:  
  
Lorem ipsum dolor
```

Если повторно вызвать функцию read(), то чтение начнется с 18-го символа.

```
file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r')  
print(file.read(17))  
print(file.read(17))  
file.close()  
# Вывод:  
  
Lorem ipsum dolor  
  
sit amet, consec
```

Функция readline

Если файл большой и Вы прочитаете его целиком, он заполнит оперативную память и может возникнуть её дефицит. Чтобы этого избежать лучше читать файл построчно. Для этого в Python есть метод `readline()`, который обеспечивает доступ к любой строке файла. Использование данного метода выглядит следующим образом:

```
file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r')
print(file.readline()) # Читает одну строку
print(file.readlines(2)) # Читает вторую строку
print(file.readlines()) # Читает все оставшиеся строки
file.close()
# Вывод:

Lorem ipsum dolor sit amet,

['consectetur adipiscing elit,\n']

['sed do eiusmod tempor incididunt\n', 'ut labore et dolore magna aliqua.']
```

Есть и другие способы.
Циклом `for`:

```
with open('C:\\Users\\VVD\\Desktop\\test.txt', 'r') as file:
    # итерация по строкам
    for line in file:
        print(line.strip())
# Вывод:

Lorem ipsum dolor sit amet,

consectetur adipiscing elit,

sed do eiusmod tempor incididunt

ut labore et dolore magna aliqua.
```

Функцией `next()`:

```
with open('C:\\Users\\VVD\\Desktop\\test.txt', 'r') as file:
    print(next(file))
    print(next(file))
    print(next(file))
    print(next(file))
# Вывод:

Lorem ipsum dolor sit amet,
```

```
consectetur adipiscing elit,  
  
sed do eiusmod tempor incididunt  
  
ut labore et dolore magna aliqua.
```

Циклом while:

```
with open('C:\\Users\\VVD\\Desktop\\test.txt', 'r') as file:  
    line = file.readline(30)  
    while line:  
        line = line.rstrip('\\n')  
        print(line)  
        line = file.readline(30) # Читаем файл кусками по 30 байт  
# Вывод:  
  
Lorem ipsum dolor sit amet,  
  
consectetur adipiscing elit,  
  
sed do eiusmod tempor incididu  
  
nt  
  
ut labore et dolore magna aliq  
  
ua.
```

Функция write

Если Вам необходимо записать что-то в файл, надо совершить следующие шаги:

- открыть файл в режиме записи;
- использовать функцию write();
- закрыть файл.

Если файла, к которому Вы обращаетесь в этом режиме, не существует, то он будет автоматически создан.

Вот как это применять:

```
file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r')  
print(file.read())  
file.close()  
file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'w')
```

```
print(file.write('Hello!'))
file.close()
file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r')
print(file.read())
file.close()
# Вывод:
```

```

Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

6

```
Hello!
```

Как Вы можете видеть из листинга выше, функция `write()` возвращает количество записанных символов.

Переименование файлов в python

Функция `rename`

Для того, чтобы поменять имя файла, Вам придётся применить функцию `rename()` из модуля `os` стандартной библиотеки Python.

Первым параметром нужно передать имя файла, который надо передать, а вторым – новое имя:

```
import os

file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r')
print(file.name)
file.close()
os.rename(file.name, 'C:\\Users\\VVD\\Desktop\\test_new.txt')
file = open('C:\\Users\\VVD\\Desktop\\test_new.txt', 'r')
print(file.name)
file.close()
# Вывод:
```

```
C:\\Users\\VVD\\Desktop\\test.txt
```

```
C:\\Users\\VVD\\Desktop\\test_new.txt
```

Текущая позиция в файлах python

Если хотите узнать текущую позицию в файле, используйте метод `tell()`. Чтоб изменить позицию – метод `seek()`.

Ghbvth:

```
file = open('C:\\Users\\VVD\\Desktop\\test.txt', 'r')
print(file.readline())
print(file.tell())
file.seek(59)
print(file.readline())
file.close()
```

Вывод:

Lorem ipsum dolor sit amet,

29

sed do eiusmod tempor incididunt

Методы файла в Python

| | |
|---|--|
| <code>file.close()</code> | закрывает открытый файл |
| <code>file.fileno()</code> | возвращает целочисленный дескриптор файла |
| <code>file.flush()</code> | очищает внутренний буфер |
| <code>file.isatty()</code> | возвращает <code>True</code> , если файл привязан к терминалу |
| <code>file.next()</code> | возвращает следующую строку файла |
| <code>file.read(n)</code> | чтение первых <code>n</code> символов файла |
| <code>file.readline()</code> | читает одну строчку строки или файла |
| <code>file.readlines()</code> | читает и возвращает список всех строк в файле |
| <code>file.seek(offset[,whence])</code> | устанавливает текущую позицию в файле |
| <code>file.seekable()</code> | проверяет, поддерживает ли файл случайный доступ. Возвращает <code>True</code> , если да |
| <code>file.tell()</code> | возвращает текущую позицию в файле |
| <code>file.truncate(n)</code> | уменьшает размер файл. Если <code>n</code> указала, то файл обрезается до <code>n</code> байт, если нет — до текущей позиции |

| | |
|--|---|
| <code>file.write(str)</code> | добавляет <u>строку</u> <code>str</code> в файл |
| <code>file.writelines(sequence)</code> | добавляет последовательность строк в файл |