Файлы Python

Файл — это всего лишь набор данных, сохраненный в виде последовательности битов на компьютере. Информация хранится в куче данных (структура данных) и имеет название «имя файла» (filename).

В Python существует два типа файлов:

1. Текстовые
2. Бинарные

Текстовые файлы

Это файлы с человекочитаемым содержимым. В них хранятся последовательности символов, которые понимает человек. Блокнот и другие стандартные редакторы умеют читать и редактировать этот тип файлов.

Текст может храниться в двух форматах: (.txt) — простой текст и (.rtf) — «формат обогащенного текста».

Бинарные файлы

В бинарных файлах данные отображаются в закодированной форме (с использованием только нулей (0) и единиц (1) вместо простых символов). В большинстве случаев это просто последовательности битов.

Они хранятся в формате .bin.

Любую операцию с файлом можно разбить на три крупных этапа:

1. Открытие файла
2. Выполнение операции (запись, чтение)
3. Закрытие файла

Открытие файла

Метод open()

В Python есть встроенная функция open(). С ее помощью можно открыть любой файл на компьютере. Технически Python создает на его основе объект.

Синтаксис следующий:

```
f = open(file_name, access_mode)
```

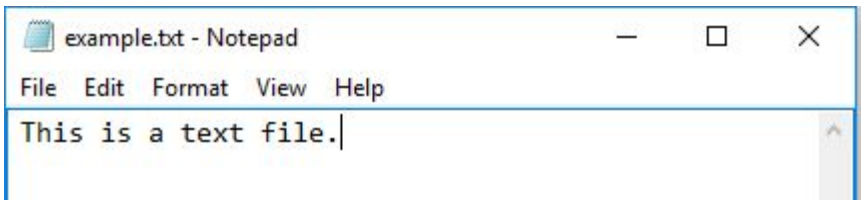
Где,

- file_name = имя открываемого файла
- access_mode = режим открытия файла. Он может быть: для чтения, записи и т. д. По умолчанию используется режим чтения (r), если другое не указано. Далее полный список режимов открытия файла

Режим	Описание
r	Только для чтения.
w	Только для записи. Создаст новый файл, если не найдет с указанным именем.
rb	Только для чтения (бинарный).
wb	Только для записи (бинарный). Создаст новый файл, если не найдет с указанным именем.
r+	Для чтения и записи.
rb+	Для чтения и записи (бинарный).
w+	Для чтения и записи. Создаст новый файл для записи, если не найдет с указанным именем.
wb+	Для чтения и записи (бинарный). Создаст новый файл для записи, если не найдет с указанным именем.
a	Откроет для добавления нового содержимого. Создаст новый файл для записи, если не найдет с указанным именем.
a+	Откроет для добавления нового содержимого. Создаст новый файл для чтения записи, если не найдет с указанным именем.
ab	Откроет для добавления нового содержимого (бинарный). Создаст новый файл для записи, если не найдет с указанным именем.
ab+	Откроет для добавления нового содержимого (бинарный). Создаст новый файл для чтения записи, если не найдет с указанным именем.

Пример

Создадим текстовый файл example.txt и сохраним его в рабочей директории.



Следующий код используется для его открытия.

```
f = open('example.txt','r')  # открыть файл из рабочей директории в режиме чтения

fp = open('C:/xyz.txt','r')  # открыть файл из любого каталога
```

В этом примере f — переменная-указатель на файл example.txt.

Следующий код используется для вывода содержимого файла и информации о нем.

```
>>> print(*f) # выводим содержимое файла

This is a text file.

>>> print(f) # выводим объект

<_io.TextIOWrapper name='example.txt' mode='r' encoding='cp1252'>
```

Стоит обратить внимание, что в Windows стандартной кодировкой является cp1252, а в Linux — utf-08.

Закрытие файла

Метод close()

После открытия файла в Python его нужно закрыть. Таким образом освобождаются ресурсы и убирается мусор. Python автоматически закрывает файл, когда объект присваивается другому файлу.

Существуют следующие способы:

Способ №1

Проще всего после открытия файла закрыть его, используя метод close().

```
f = open('example.txt','r')

# работа с файлом
```

```
f.close()
```

После закрытия этот файл нельзя будет использовать до тех пор, пока заново его не открыть.

Способ №2

Также можно написать try/finally, которое гарантирует, что если после открытия файла операции с ним приводят к исключениям, он закроется автоматически.

Без него программа завершается некорректно.

Вот как сделать это исключение:

```
f = open('example.txt','r')
try:
  # работа с файлом
finally:
  f.close()
```

Файл нужно открыть до инструкции try, потому что если инструкция open сама по себе вызовет ошибку, то файл не будет открываться для последующего закрытия.

Этот метод гарантирует, что если операции над файлом вызовут исключения, то он закроется до того как программа остановится.

Способ №3

Инструкция with

Еще один подход — использовать инструкцию with, которая упрощает обработку исключений с помощью инкапсуляции начальных операций, а также задач по закрытию и очистке.

В таком случае инструкция close не нужна, потому что with автоматически закроет файл.

Вот как это реализовать в коде.

```
with open('example.txt') as f:
```

```
# работа с файлом
```

Чтение и запись файлов в Python

В Python файлы можно читать или записывать информацию в них с помощью соответствующих режимов.

Функция read()

Функция read() используется для чтения содержимого файла после открытия его в режиме чтения (r).

Синтаксис

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

Где,

- file = объект файла
- size = количество символов, которые нужно прочитать. Если не указать, то файл прочитается целиком.

Пример

```
>>> f = open('example.txt','r')

>>> f.read(7)  # чтение 7 символов из example.txt

'This is '
```

Интерпретатор прочитал 7 символов файла и если снова использовать функцию read(), то чтение начнется с 8-го символа.

```
>>> f.read(7)  # чтение следующих 7 символов

' a text'
```

Функция readline()

Функция readline() используется для построчного чтения содержимого файла. Она используется для крупных файлов. С ее помощью можно получать доступ к любой строке в любой момент.

Пример

Создадим файл test.txt с нескольким строками:

This is line1.

This is line2.

This is line3.

Посмотрим, как функция readline() работает в test.txt.

```
>>> x = open('test.txt','r')
>>> x.readline()  # прочитать первую строку
This is line1.
>>> x.readline(2)  # прочитать вторую строку
This is line2.
>>> x.readlines()  # прочитать все строки
['This is line1.','This is line2.','This is line3.']
```

Обратите внимание, как в последнем случае строки отделены друг от друга.

Функция write()

Функция write() используется для записи в файлы Python, открытые в режиме записи.

Если пытаться открыть файл, которого не существует, в этом режиме, тогда будет создан новый.

Синтаксис

```
file.write(string)
```

Пример

Предположим, файла xyz.txt не существует. Он будет создан при попытке открыть его в режиме чтения.

```
>>> f = open('xyz.txt','w')  # открытие в режиме записи
>>> f.write('Hello \n World')  # запись Hello World в файл
```

```
Hello

World

>>> f.close()  # закрытие файла
```

Переименование файлов в Python

Функция rename()

Функция rename() используется для переименовывания файлов в Python. Для ее использования сперва нужно импортировать **модуль os**.

Синтаксис следующий.

```
import os

os.rename(src,dest)
```

Где,

- src = файл, который нужно переименовать
- dest = новое имя файла

Пример

```
>>> import os

>>> # переименование xyz.txt в abc.txt

>>> os.rename("xyz.txt","abc.txt")
```

Текущая позиция в файлах Python

В Python возможно узнать текущую позицию в файле с помощью функции tell(). Таким же образом можно изменить текущую позицию командой seek().

Пример

```
>>> f = open('example.txt')  # example.txt, который мы создали ранее

>>> f.read(4)  # давайте сначала перейдем к 4-й позиции

This
```

```
>>> f.tell()  # возвращает текущую позицию
4
>>> f.seek(0,0)  # вернем положение на 0 снова
```

Методы файла в Python

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

Задания для выполнения

1. С помощью генератора псевдослучайных чисел создайте список из 10 элементов от 0 до 9. Каждое число списка сохраните в текстовый файл, причем каждое число с новой строки.

2. Сделайте копию файла из задания 1. Выполните чтение скопированного файла. Каждую прочитанную цифру замените на слово (1 – один, 2 – два, и т.д.). Сохраните в новый текстовый документ.
3. Напишите программу на Python для построчного чтения файла и сохранения его в списке.
4. Напишите программу на Python для объединения каждой строки из первого файла с соответствующей строкой во втором файле
5. Напишите программу на python, чтобы найти самые длинные слова
6. Создайте 10 текстовых файлов с разным содержимым. Сохраните в словарь информацию о названии файла и количестве слов в каждом из них. Отсортируйте словарь по количеству слов в каждом файле. Переименуйте каждый текстовый документ, в качестве названия – число слов в файле.
7. Создайте папку с 20 файлами разного формата (.txt, .jpg, .png, .mp3 и т.д.) Выполните сортировку файлов по форматам. Т.е. для каждого формата данных создать папки, назвать их в соответствии с форматом, переместить туда файлы.
8. В каталоге с 10 текстовыми файлами разного объема, найти файл с самым большим весом (в байтах). Определите дату и время формирования этого файла. Переименуйте файл в формате ДД_ММ_ГГГГ_ЧЧ_ММ_СС (день, месяц, год, часы, минуты, секунды)
9. Сохраните в файлы решения задач из предыдущей практической работы. Название файлов – номер задачи. Создайте словарь в котором будет содержаться название фала и его объем.