



Файловая система





План занятия

1

Файлы Python

2

Открытие файла

3

Заккрытие файла

4

Чтение и запись
файлов в Python

5

Работа с библиотекой os



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:

- Текстовые
- Бинарные

Текстовые файлы

Это файлы с 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».

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:

- Открытие файла
- Выполнение операции (запись, чтение)
- Закрытие 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.

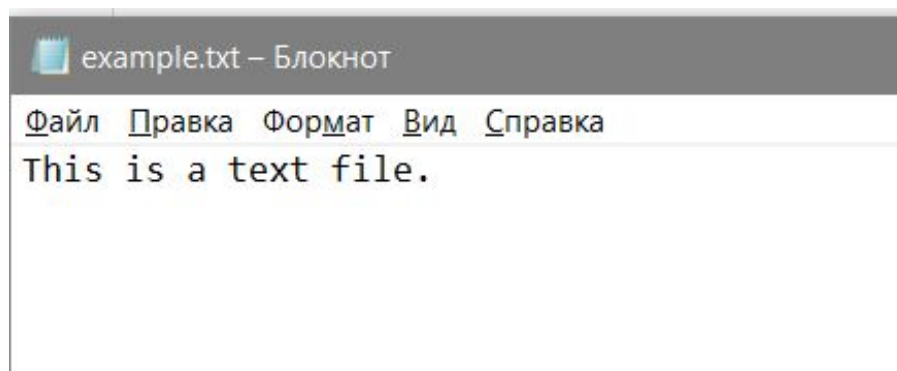
Он может быть: для чтения, записи и т. д. По умолчанию используется режим чтения (**r**), если другое не указано.

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
```


Пример

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  
print(f) # выводим объект
```

```
This is a text file.  
<_io.TextIOWrapper name='example.txt' mode='r' encoding='cp1251'>  
  
Process finished with exit code 0
```

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

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:
    print(*f) # 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.

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:  
    print(*f)  # работа с файлом
```

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 прочесть. Если не указать, то файл прочитается целиком.

Пример

```
f = open('example.txt', 'r')  
  
print(f.read(7)) # чтение 7 символов из example.txt
```

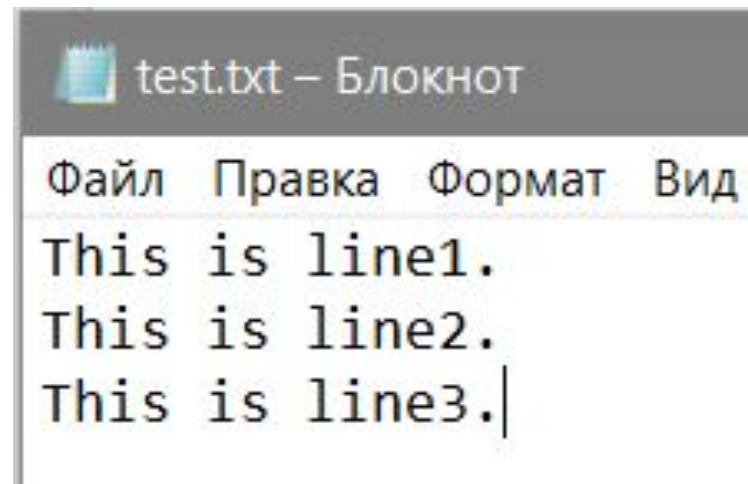
Интерпретатор прочитал 7 символов файла и если снова использовать функцию `read()`, то чтение начнется с 8-го символа.

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 несколькими строками:



```
test.txt – Блокнот
Файл  Правка  Формат  Вид
This is line1.
This is line2.
This is line3.|
```

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

```
x = open('test.txt', 'r')  
  
print(x.readline()) # прочитать первую строку  
print(x.readline()) # прочитать вторую строку
```

```
x = open('test.txt', 'r')  
  
print(x.readlines()) # прочитать все строки
```


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 \nWorld') # запись Hello World в файл
f.close() # закрытие файла
```

Функция `rename()`

Функция `rename()` используется для переименования 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

os.rename("xyz.txt", "abc.txt") # переименование xyz.txt в abc.txt
```

Для получения текущего рабочего каталога используется `os.getcwd()`:

```
import os

print("Текущая деректория:", os.getcwd()) # вывести текущую директорию
```

`os.getcwd()` возвращает строку в Юникоде, представляющую текущий рабочий каталог.

Для создания папки/каталога в любой операционной системе нужна следующая команда:

```
import os

print("Текущая директория:", os.getcwd()) # вывести текущую директорию

os.mkdir("folder") # создать пустой каталог (папку)
```

После ее выполнения в текущем рабочем каталоге тут же появится новая папка с названием «folder».

Если запустить ее еще раз, будет вызвана ошибка `FileExistsError`, потому что такая папка уже есть. Для решения проблемы нужно запускать команду только в том случае, если каталога с таким же именем нет.

Менять директории довольно просто. Прделаем это с только что созданным:

```
import os

# изменение текущего каталога на 'folder'
os.chdir("folder")

# вывод текущей папки
print("Текущая директория изменилась на folder:", os.getcwd())
```

Предположим, вы хотите создать не только одну папку, но и несколько вложенных:

```
import os

os.chdir("folder") # изменение текущего каталога на 'folder'

os.chdir("..") # вернуться в предыдущую директорию

os.makedirs("nested1/nested2/nested3") # сделать несколько вложенных папок
```

Это создаст три папки рекурсивно.

Удалим созданный файл:

```
import os

# удалить этот файл
os.remove("folder/renamed-text.txt")
```

С помощью функции `os.rmdir()` можно удалить указанную папку:

```
import os

# удалить папку
os.rmdir("folder")
```

Для удаления каталогов рекурсивно необходимо использовать `os.removedirs()`:

```
import os

# удалить вложенные папки
os.removedirs("nested1/nested2/nested3")
```

Это удалит только пустые каталоги.

Задание №1



В файле, в одну строку записаны слова и числа через пробел или _
найти сумму всех чисел.

```
with open('task_1.txt') as f:
    s = f.readlines()
    print(s)
for i in s:
    i = i.replace('_', ' ')
    l = i.split(' ')
    print(l)
    sum = 0
    for i in l:
        if i.isdigit():
            i = int(i)
            sum += i
    print(sum)
```

 task_1.txt – Блокнот

Файл Правка Формат Вид Справка

123 435 432 asd 123_df_34

Задание №2



Файл содержит числа и буквы. Каждый записан в отдельной строке. Нужно считать содержимое в список так, чтобы сначала шли числа по возрастанию, а потом слова по возрастанию их длины.

```
f = open('task_2.txt')
b = []
n = []
s = f.readlines()
print(s)
for i in s:
    i = i[:-1]
    if i.isalpha():
        i = str(i)
        b.append(i)
    elif i.isdigit():
        i = int(i)
        n.append(i)
print(b)
print(n)
n.sort()
b.sort()
mas = n + b
print(mas)
```

task_2.txt – Блокнот

Файл Правка Формат Вид Спра

python
java
с
2
1
5
3

Задание №3



Создать текстовый файл, записать в него построчно данные, которые вводит пользователь. Окончанием ввода пусть служит пустая строка.



```
fname = input('Файл: ')\nf = open(fname, 'w')\nwhile True:\n    s = input()\n    if s == '':\n        break\n    f.write(s + '\\n')\nf.close()
```

Задание №4



В текстовом файле посчитать количество строк, а также для каждой отдельной строки определить количество в ней символов.

```
f = open('task_4.txt')
line = 0
for i in f:
    line += 1
    print(i, len(i), 'симв.')
print(line, 'стр.')
f.close()
```

 task_4.txt – Блокнот

Файл Правка Формат Вид Справка

asdasd dghfg
dff xxvx df



Домашнее задание

Есть массив состоящий из слов и чисел, нужно записать в файл сначала слова в порядке их длины, а после слов цифры в порядке возрастания