Работа с файлами

Date: 07/04/2024

Tochilkina Ksenya Stanislavovna

Работа с файлами

Сегодня научимся создавать, читать, а так же записывать файлы.

В языке программирования python существуют множества функция для работы с файлами, однако стоит всегда помнить некоторые нюансы:
1. Любой файл, с которым вы хотите работать, нужно изначально **“открыть”**. Для этого существует определенная функция.
2. После открытия файла, выполняем все необходимые операции с ним, и затем нужно **ВСЕГДА ЗАКРЫВАТЬ** файл. И это очень важный момент, так как многие забывают закрывать файл, и у многих происходит утечка памяти. Если вы не возьмете в привычку закрывать файлы, и, допустим, в проекте у вас будут открыты несколько файлов и вы их не будете закрывать, то ваш проект будет уже достаточно перегружен, так как программа будет тратить ресурсы на обработку каждого из открытых файлов.

Открытие файлов

Чтобы открыть файл, нам необходимо использовать функцию - **open()**. Эта функция принимает несколько параметров:
Первый параметр: указывает на то, какой конкретно файл мы будет открывать. Если файл, находится у вас в проекте, то достаточно прописать его название. Однако, если он хранится в любом другом месте, то необходимо указать весь его путь.

```
file = open('text.txt')
```

Хочется отметить, что если указанного файла не будет у вас в проекте, он будет создан автоматически. Так же, если вы хотите чтобы файл создался в какой-то конкретной папке проекта, то необходимо указать название папки (уже существующей), а затем название файла.

```
file = open('data/text.txt')
```

Второй параметр: указываем способ открытия файла (доступ открытия). Существуют разные способы открытия файлов: можно открыть файл для чтения, для записи, можно открыть файл просто для его создания и т.д. Вот возможные варианты открытия:

Режим	Обозначение
'r'	открытие на чтение (является значением по умолчанию).
'w'	открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
'x'	открытие на запись, если файла не существует, иначе исключение.
'a'	открытие на дозапись, информация добавляется в конец файла.
'b'	открытие в двоичном режиме.
't'	открытие в текстовом режиме (является значением по умолчанию).
'+'	открытие на чтение и запись

На данный момент, будем использовать режим открытия ‘w’ - открытие для дальнейшей записи информации внутрь файла.

```
file = open('data/text.txt', 'w')
```

Следующее, что я сразу предлагаю Вам делать (!!!всегда!!!) - это закрывать файл. Для этого необходимо использовать функцию **close()**. Делайте это сразу после открытия файла, а уже между этими выражениями записывайте манипуляции с файлами.

```
file = open('data/text.txt', 'w')


file.close()
```

Запись данных в файл

Для того, чтобы записать информацию внутрь файла необходимо использовать метод **write()** - этот метод позволяет занести некоторую строку/информацию внутрь файла.

```
file = open('data/text.txt', 'w')

file.write('Hello')

file.close()
```

При этом, при запуске программы в консоли будет пусто, однако в папке “data” уже автоматически создался файл “text.txt”, в котором при открытии мы увидим текст “Hello”.

Давайте попробуем добавить еще текст, например:

```
file = open('data/text.txt', 'w')

file.write('Hello')
file.write('!!!')

file.close()
```

Можно сделать вывод, что текст “!!!” будет выведен на новой строке, однако это не так. Не смотря на то, что мы вызвали два метода **write()** перехода на новую строку у нас не произошло. Так происходит по той причине, что в отличии от метода **print()** у нас в конце нет автоматического перехода на новую строку. Однако, если он нам необходим, мы можем прописать его самостоятельно с помощью “\n”.

```
file.write('Hello\n')
file.write('!!!')
```

Теперь у нас восклицательные знаки запишутся с новой строки. При этом, сколько бы раз мы не запускали проект, у нас результат в текстовом файле будет выглядеть всегда одинаково. То есть, при каждом запуске проекта, весь текст в файле будет перезаписываться, а не добавляться к уже существующему.

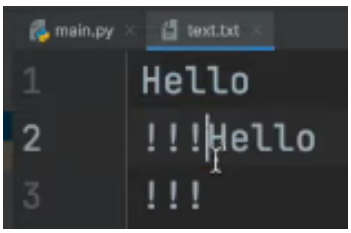
Если же мы хотим к текущему тексту, добавить новый, не перезаписывая весь файл, можно обратиться к режиму открытия “a”(append) - позволяет добавить информацию В КОНЕЦ документа.

```
file = open('data/text.txt', 'a')

file.write('Hello\n')
file.write('!!!')

file.close()
```

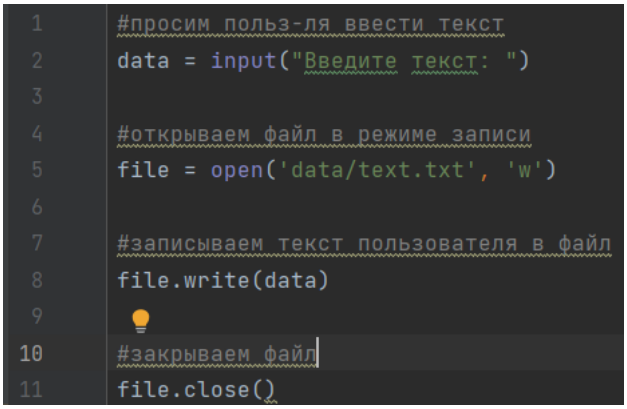
Работает он в целом таким же образом, только результат в файле разный.
- ПЕРВАЯ КАРТИНКА скриншот выполнения программы с режимом “w”.
- ВТОРАЯ КАРТИНКА скриншот выполнения программы с режимом “a”.



МИНИ-ВЫВОД: если необходимо *перезаписать информацию*, используйте режим доступа *“w”*, однако, если необходимо *добавить информацию* к уже существующему тексту в файле, используйте режим *“a”*.

Пользовательские данные

Рассмотрим ситуацию, когда необходимо чтобы пользователь вводил информацию, например, в консоль, а мы эту информацию уже записывали в файл.



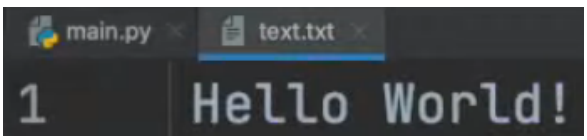
```
#просим польз-ля ввести текст
data = input("Введите текст: ")

#открываем файл в режиме записи
file = open('data/text.txt', 'w')

#записываем текст пользователя в файл
file.write(data)

#закрываем файл
file.close()
```

При запуске, в консоли необходимо ввести текст, например “Hello world!” и он корректно перезапишется в нашем текстовом файле.



Считывания данных из файла

Для считывания данных, нам, по сути, необходимо использовать всю туже логику. Единственная разница - это вид открытия файла “r” и вместо функции записи, используем функцию для чтения данных - **read().**

```
#открываем файл в режиме чтения
file = open('data/text.txt', 'r')
print(file.read())
file.close()
```

При запуске программы, мы видим, что метод **read()** выводит в консоль всю информацию файла. Так происходит, если мы не добавляем в нее никаких параметров. Но внутри мы можем прописать следующее:

```
file = open('data/text.txt', 'r')

print(file.read(4))

file.close()
```

Цифра 4 обозначает число символом, которое будет выведено от начала строки в файле.