Назначение лекции

Вы узнаете, как выполнять запись и чтение текстовых файлов в Python.

Запись в файл дает возможность сохранять данные расчетов для дальнейшего использования. Если для хранения использовать оперативную память, то они будут сохранятся лишь кратковременно.

Файл – это именованная область во внешней памяти.

В современной интерпретации внутренняя структура файла для операционной системы не имеет значения.

Но это было не всегда. В первых файловых системах файлы были структурированы (состояли из отдельных записей), но в последствии от этого отказались.

Файлы различаются атрибутами (свойствами). Эти атрибуты зависят от операционной системы.

Одни из главных атрибутов – это **имя** файла и **тип**. Распространенные типы файла: каталог, ярлык, специальный файл, а также файлы для конкретного приложения.

Остальные атрибуты также зависят от операционной системы, например, время последней записи, размер, права доступа и др.

Для операционной системы Microsoft Windows имеет значение, так называемое, расширение файла, например, расширение .txt дает возможность работать с файлом в любом текстовом редакторе. Расширение .docx для работы в Microsoft Word и т.д.

Работа с файлами обеспечивается файловой системой, которая входит в состав операционной системы. Разные операционные системы обеспечивают работу с разными файловыми системами (одной или несколькими)

Microsoft Windows – файловая система NTFS и FAT32

Unix – Unix File System (UFS)

Linux - **Extended Filesystem (EXT), XFS**

**Но в общем случае, принципы работы с файлами примерно одинаковы:**

1. **открыть файл или создать,**
2. **работать с файлом,**
3. **закрыть файл.**

**После работы с файлом закрывать файл надо обязательно, т.к. изменения не сохранятся.**

**Далее рассмотрим основные принципы работы с текстовыми файлами в Python.**

**Существуют несколько основных режимов работы с файлами.**

|  |  |
| --- | --- |
| **Режим** | **Описание** |
| **w** | **Режим записи**: используется для создания нового файла. Существующий файл с заданным именем стирается, и вместо него создается новый файл |
| **r** | **Режим чтения**: используется в том случае, если существующий файл используется только для чтения, а запись в него не выполняется |
| **a** | Режим присоединения: используется для добавления новых данных в конец файла. Если файла нет , то он создается |

Текстовые файлы могут использоваться только для записи, чтения и присоединения данных. Природа текстовых файлов такова, что данные, записанные в файл, не так просто удалить или изменить; приходится перезаписывать весь файл или создавать новый для хранения обновленных данных.

Если вы хотите иметь возможность изменять отдельные элементы после того, как файл будет создан, лучше воспользоваться файлом с расширением .csv или базой данных SQL. Эти возможности будут рассмотрены далее.

**file = open("MyFile.txt", "w")**

**file.write("Бишкек\n")**

**file.write("Ош\n")**

**file.write("Джалал-Абат\n")**

**file.write("Талас\n")**

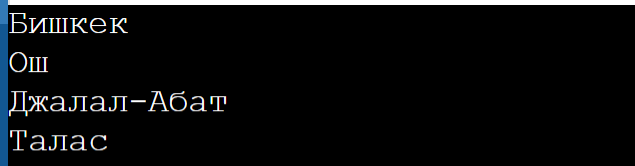
**file.close()**

Создает файл с именем **MyFile.txt**. Если файл с таким именем уже существует, то он заменяется новым пустым файлом. В него добавляются четыре строки данных (\n вставляет разрыв строки после каждой записи). Затем файл закрывается, чтобы изменения в текстовом файле были сохранены.

**file = open("MyFile.txt", "r")**

**print(file.read())**

Открывает файл **MyFile.txt** в режиме чтения и выводит все содержимое файла:



**file = open("MyFile.txt ", "a")**

**file.write("Tокмак\n")**

**file.close()**

Открывает файл **MyFile.txt** в режиме присоединения, добавляет новую строку и закрывает файл. Если строка **file.close()** отсутствует, то изменения не будут сохранены в текстовом файле.

**file = open("Numbers.txt", "w")**

**file.write("4, ")**

**file.write("6, ")**

**file.write("10, ")**

**file.write("8, ")**

**file.write("5, ")**

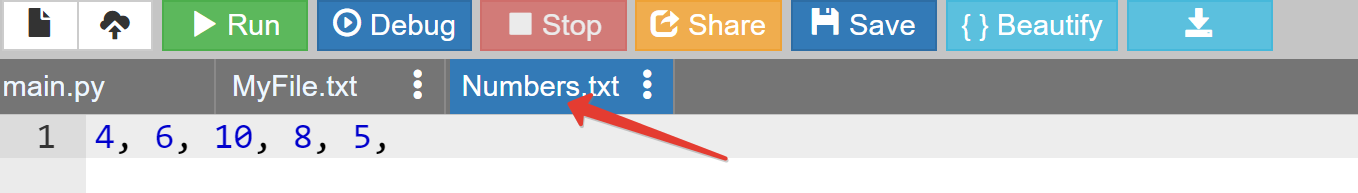
**file.close()**

**file = open("Numbers.txt", "r")**

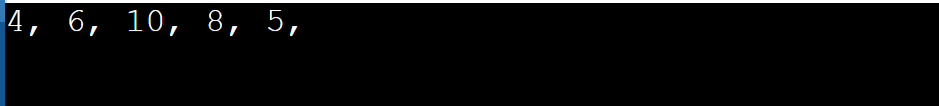
**print(file.read())**

**file.close()**

Открывает файл **Numbers.txt** в режиме записи, добавляет числа с запятой и пробелом, но без «return» и закрывает файл. Тут же открываем файл на чтение и печатаем. После чего опять закрываем. Содержимое файла:



И результат печати:



**file = open("Names.txt", "w")**

**file.write("Александр\n")**

**file.write("Эмир\n")**

**file.write("Айжан\n")**

**file.write("Элхан\n")**

**file.write("Алия\n")**

**file.close()**

В файл Names.txt записываются имена, построчно.

**file = open("Names.txt", "a")**

**newname = input("Enter a new name: ")**

**file.write(newname + "\n")**

**file.close()**

**file = open("Names.txt", "r")**

**print (file.read())**

**file.close()**

В файл Names.txt записываются в конец имя, которое ввел пользователь. После этого файл читается и печатается.

Далее разберем две важные ситуации, которые могут встретится при работе с файлами.

Первая – признак конца файла. Часто данные файла читаются и обрабатываются построчно, до тех пор пока не встретится конец файла. Как узнать что встретился конец файла?

**file = open("Names.txt", "r", encoding="utf-8")  
while True:  
 line = file.readline()  
 if line=='':  
 break  
 else:  
 print(line, end='')  
file.close()**

Открыт файл Name.txt для чтения русских букв и в цикле читаются и печатаются строки из этого файла, пока не встретится конец файла **if line!='':**

**file=open("Names.txt", "r", encoding="utf-8")  
for line in file:  
 print(line, end='')**

В цикле for читаются строки из файла, пока не встретится конец файла.

**file = open("Names.txt", "r", encoding="utf-8")  
content = file.read()  
if content.strip():  
 print('Файл не пуст')  
else:  
 print('Файл пуст')  
file.close()**

Файл Name.txt читается и через функцию strip удаляются все пробелы. Если полученная строка пуста, то **content.strip()** принимает значение True и следовательно файл пуст.

Итак были рассмотрены все самые важные варианты работы с текстовыми файлами. Далее рассмотрим несколько рекомендаций.

Когда использовать режим **w** и когда **a**?

Если используется режим **w**, то при каждом запуске программы файл будет перезаписываться заново, а в режими **a**, старое содержание остается и добавляются новые строки.

При объявлении файла рекомендуется писать код следующим образом:

**with open("Names.txt", "r", encoding="utf-8") as file:**

**for line in file:**

**print(line, end='')**

или

**with open("Names.txt", "w") as file:**

**file.write("Александр\n")**

**file.write("Эмир\n")**

**file.write("Айжан\n")**

**file.write("Элхан\n")**

**file.write("Алия\n")**

конструкция **with open(...) as file**, которая автоматически открывает файл "**Names.txt**" для записи и закрывает его после завершения работы блока кода.

После выполнения блока кода, связанного с конструкцией with, файл будет автоматически закрыт, даже если возникнут аварии в процессе работы. Это гарантирует правильную запись данных в файл и предотвращает утечку ресурсов.

Конструкция with open(...) as file рекомендуется для работы с файлами в Python, так как она облегчает управление файловыми объектами и обеспечивает корректное закрытие файлов.

Конец лекции