**Работа с файлами в Python**

Python - это высокоуровневый интерпретируемый язык программирования, который позволяет писать код более читаемо и понятно. Он широко используется во многих областях программирования, таких как веб-разработка, наука о данных, искусственный интеллект и многие другие.

Python имеет простой и понятный синтаксис, что делает его отличным выбором как для начинающих программистов, так и для опытных разработчиков. Он поддерживает различные парадигмы программирования, такие как процедурное, объектно-ориентированное и функциональное программирование.

Python также удобен для работы с различными типами данных, имеет обширную стандартную библиотеку и множество сторонних модулей, что позволяет легко решать различные задачи.

Благодаря своей популярности и удобству использования, Python стал одним из самых популярных языков программирования в мире.

**Что такое файл**

На любом компьютере и в любой операционной системе есть файлы — область данных со своим именем, хранящаяся на носителе. Их принято считать базовыми объектами, из которых складываются директории. В Python с помощью файлов можно сохранять результат работы программы или получать из них данные для обработки в коде.

С любыми файлами можно производить действия, которые делятся на три группы:

* открытие;
* операции чтения из файла и записи в файл;
* закрытие.

Дело в том, что Python считает некорректным стандартный для Windows формат: если указать путь к файлу в привычном виде 'C:\Users\User\Python\', интерпретатор вернет ошибку. Лучше всего указывать путь с помощью r-строк (r’) или с экранированием слэшей(\\). Иногда путь указывают с обратными слэшами.(/)

**Открытие файла на чтение и запись в Python**

На компьютере файл надо открыть перед тем, как вносить в него изменения. Такое же правила действует и для Python: нужно открыть файл на чтение и запись. Без этапа открытия нельзя читать содержимое или изменять его. Открыть файл можно с помощью функции **«open()»**

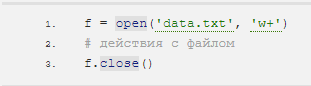


В функцию в качестве аргументов требуется передать путь файлу (file) и выбрать режим работы **«(mode)»**. По умолчанию Python выбирает значение **«rt»**, но доступны и другие режимы:

|  |  |
| --- | --- |
| **Аргумент mode** | **Как работает** |
| r | Чтение из файла |
| t | Открыть как текстовый файл |
| w | Запись в файл и создание файла, если его не существует |
| x | Запись в файл и вызовы исключения, если файла не существует |
| b | Открыть как двоичный файл |
| a | Запись в файл путем добавления новых значений в конец |
| + | Работа в режиме чтения и записи |

Можно комбинировать режимы и использовать сразу несколько. К примеру, по умолчанию функция **«open()»** принимает аргумент **«rt»**. Это значит, что файл будет открыт в виде текстового документа для чтения. С помощью **«r+»** и **«w+»** можно открыть файл сразу для чтения и записи на Python. В первом случае несуществующие файлы будут создаваться, а во втором такое действие спровоцирует вызов исключения.

После окончания работы файлы надо закрывать — так же, как мы это делаем на компьютере. Для этого используется функция **«close()»** после кода взаимодействия с файлом:



Также файлы можно открывать с помощью менеджера контекста «**with»**. В этом случае файл автоматически закроется, когда работа с ним завершится:



**Чтение файла**

Чтение из файла Python можно реализовать с помощью функции **«read()»**, если он открыт в режиме чтения **«r»**

Синтаксис функции выглядит следующим образом:



В строке выше **«file»** обозначает объект, с которым работаем, а «size»— количество символов для чтения. Если ничего не указать в качестве аргумента функции **«read»()**, то получится прочитать сразу все содержимое документа.

Функция **readline()** в языке программирования Python используется для чтения одной строку из открытого файлового объекта. Она считывает одну строку из файла, включая символы новой строки, и возвращает эту строку как результат.

Функция **list()** в Python используется для создания списка из итерируемого объекта, такого как строка, кортеж, множество или словарь.

**Запись в файл на Python**

Для записи файла в Python используется функция **«write()»**. В качестве аргумента ей следует передать строку, содержимое которой будет записано:



В файл можно записать сразу список строк. Для этого применяется метод «**writelines**()», которому в качестве аргумента надо передать сам список. Метод «**writelines**()»не добавляет разделители строк автоматически. Нужно сделать это вручную, прописав **«\n»** каждому элементу списка:

**Управление указателем**

В Python есть возможность явно задать позицию указателя в файле - для этого используется метод «**seek()»**.Узнать текущую позицию можно с помощью метода **«tell()»**

Метод **«seek()»** в Python используется для перемещения указателя текущей позиции в файле. Он принимает два аргумента: **«offset»** (смещение) и **«whence»** (откуда).

Аргумент **«offset»** указывает на количество байт, на которое нужно переместить указатель, а аргумент **«whence»** указывает относительно чего нужно делать смещение: с начала файла (0), с текущей позиции (1) или с конца файла (2). Узнать байт позиции указателя можно с помощью метода **«tell()»**.

**Работа с файловой системой**

В Python можно работать не только с конкретным файлом, но и со всей системой. Перемещаться между каталогами, создавать новые файлы и переименовывать существующие.

Для работы с файловой системой на Python используют встроенная библиотека OS. Ее необходимо отдельно импортировать в код проекта, чтобы получить доступа к ее методам:



С помощью метода **«getcwd()»** мы можем узнать текущий каталог

С помощью метода **«listdir()»** может получить список с содержимым каталога

Для создания новых директорий файлов в Python используют метод **«mkdir()»** . В качестве аргумента ему надо передать полный путь, включая название новой директории.

С помощью Python можно переименовывать файлы и директории. Для этого используется метода **«rename()»** которому необходимо передать в качестве аргументов путь к старому и новым файлам.

Для удаления пустых директорий файлов в Python можно использовать метод **«rmdir()»** , а для удаления директории со всем содержимым **«shutil.rmtree()»**