**Работа с файлами**

**Открытие и закрытие файлов**

Python поддерживает множество различных типов файлов, но условно их можно разделить на два виде: текстовые и бинарные. Текстовые файлы - это к примеру файлы с расширением cvs, txt, html, в общем любые файлы, которые сохраняют информацию в текстовом виде. Бинарные файлы - это изображения, аудио и видеофайлы и т.д. В зависимости от типа файла работа с ним может немного отличаться.

При работе с файлами необходимо соблюдать некоторую последовательность операций:

1. Открытие файла с помощью метода **open()**
2. Чтение файла с помощью метода **read()** или запись в файл посредством метода **write()**
3. Закрытие файла методом **close()**

### Открытие и закрытие файла

Чтобы начать работу с файлом, его надо открыть с помощью функции **open()**, которая имеет следующее формальное определение:

|  |  |
| --- | --- |
| 1 | open(file, mode) |

Первый параметр функции представляет путь к файлу. Путь файла может быть абсолютным, то есть начинаться с буквы диска, например, *C://somedir/somefile.txt*. Либо можно быть относительным, например, *somedir/somefile.txt* - в этом случае поиск файла будет идти относительно расположения запущенного скрипта Python.

Второй передаваемый аргумент - mode устанавливает режим открытия файла в зависимости от того, что мы собираемся с ним делать. Существует 4 общих режима:

* **r** (Read). Файл открывается для чтения. Если файл не найден, то генерируется исключение FileNotFoundError
* **w** (Write). Файл открывается для записи. Если файл отсутствует, то он создается. Если подобный файл уже есть, то он создается заново, и соответственно старые данные в нем стираются.
* **a** (Append). Файл открывается для дозаписи. Если файл отсутствует, то он создается. Если подобный файл уже есть, то данные записываются в его конец.
* **b** (Binary). Используется для работы с бинарными файлами. Применяется вместе с другими режимами - w или r, например, **rb** (чтение бинарных файлов) и **wb** (запись бинарных файлов).
* **r+**. Файл открывается одновременно для чтения и записи. Если файл не найден, то генерируется исключение FileNotFoundError
* **w+**. Файл открывается одновременно для чтения и записи. Если файл не существует, то он автоматически создается. Если файл существует, то он перезаписывается
* **a+**. Файл открывается одновременно для чтения и записи. Если файл не существует, то он автоматически создается. Если файл существует, то данные добавляются в конец файла

Например, открытие текстового файла на запись:

myfile = open("hello.txt", "w")

Или открытие бинарного файла на чтение

myfile = open("image.png", "rb")

После завершения работы с файлом его обязательно нужно закрыть методом close(). Данный метод освободит все связанные с файлом используемые ресурсы.

Например, откроем для записи текстовый файл "hello.txt":

myfile = open("hello.txt", "w")

myfile.close()

При открытии файла или в процессе работы с ним мы можем столкнуться с различными исключениями, например, к нему нет доступа и т.д. В этом случае программа выпадет в ошибку, а ее выполнение не дойдет до вызова метода close, и соответственно файл не будет закрыт.

В этом случае мы можем обрабатывать исключения:

try:

    myfile = open("hello.txt", "w")

    try:

        print("Работа с файлом")

    finally:

        myfile.close()

except Exception as ex:

    print(ex)

В данном случае вся работа с файлом идет во вложенном блоке try. И если вдруг возникнет какое-либо исключение, то в любом случае в блоке finally файл будет закрыт.

Однако есть и более удобная конструкция - конструкция **with**:

with open(file, mode) as myfile:

    инструкции

Эта конструкция определяет для открытого файла переменную myfile и выполняет набор инструкций. После их выполнения файл автоматически закрывается. Даже если при выполнении инструкций в блоке with возникнут какие-либо исключения, то файл все равно закрывается.

Так, перепишем предыдущий пример:

with open("hello.txt", "w") as myfile:

    print("Работа с файлом myfile")

## Текстовые файлы

### Запись в текстовый файл

Чтобы открыть текстовый файл на запись, необходимо применить режим **w** (перезапись), **a**(дозапись) или **r+**/**w+** (чтение и запись) . Затем для записи применяется метод **write(str)**, в который передается записываемая строка. Стоит отметить, что записывается именно строка, поэтому, если нужно записать числа, данные других типов, то их предварительно нужно конвертировать в строку.

Запишем некоторую информацию в файл "hello.txt":

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

    file.write("hello world")

print("Файл записан")

Если мы откроем папку, в которой находится текущий скрипт Python, то увидем там файл "hello.txt". Этот файл можно открыть в любом текстовом редакторе и при желании изменить.

Теперь дозапишем в этот файл еще одну строку:

with open("hello.txt", "a") as file:

    file.write("\nhello work")

print("Файл изменен")

Дозапись выглядит как добавление строки к последнему символу в файле, поэтому, если необходимо сделать запись с новой строки, то можно использовать эскейп-последовательность "\n". В итоге файл hello.txt будет иметь следующее содержимое:

hello world

hello work

Для записи списка строк имеется метод **writelines()**. В качестве параметра он принимает список строк:

lines = ["Hello Word\n", "Hello Work\n", "Hello World\n"]

with open("hello2.txt", "a") as file:

    file.writelines(lines)

print("Список строк записан в файл")

Стоит учитывать, что, если мы хотим, чтобы каждая строка из списка располагалась на отдельной строке в файле, то строка в списке должна заканчиваться на символ "\n".

Еще один способ записи в файл представляет стандартный метод **print()**, который применяется для вывода данных на консоль:

with open("hello.txt", "a") as myfile:

    print("\nhello metanit.com", file=myfile)

Для вывода данных в файл в метод print в качестве второго параметра передается название файла через параметр file. А первый параметр представляет записываемую в файл строку.

### Чтение файла

Для чтения файла он открывается с режимом **r** (Read), и затем мы можем считать его содержимое различными методами:

* **readline()**: считывает одну строку из файла
* **read()**: считывает все содержимое файла в одну строку
* **readlines()**: считывает все строки файла в список

Например, считаем выше записанный файл построчно:

with open("hello.txt", "r") as file:

    for line in file:

        print(line)

Несмотря на то, что мы явно не применяем метод readline() для чтения каждой строки, но в при переборе файла этот метод автоматически вызывается для получения каждой новой строки. Поэтому в цикле вручную нет смысла вызывать метод readline. Консольный вывод:

hello world

hello work

hello metanit.com

Однако поскольку функция print неявно добавляет к каждой строке перевод строки "\n", а строки в записанном файлетакже разделяются символом перевода строки "\n", то получается, что каждая строка оканчивается двойным "\n". и чтобы исключить излишнего переноса на другую строку в функцию print можно передать значение end="".

with open("hello.txt", "r") as file:

    for line in file:

        print(line, end="")

Теперь явным образом вызовем метод readline() для чтения отдельных строк:

with open("hello.txt", "r") as file:

    str1 = file.readline()      # считываем первую строку

    print(str1, end="")

    str2 = file.readline()      # считываем вторую строку

    print(str2)

Консольный вывод:

hello world

hello work

Метод readline можно использовать для построчного считывания файла в цикле while:

with open("hello.txt", "r") as file:

    line = file.readline()

    while line:

        print(line, end="")

        line = file.readline()

Если файл небольшой, то его можно разом считать с помощью метода **read()**:

with open("hello.txt", "r") as file:

    content = file.read()

    print(content)

И также применим метод **readlines()** для считывания всего файла в список строк:

with open("hello.txt", "r") as file:

    contents = file.readlines()

    str1 = contents[0]

    str2 = contents[1]

    print(str1, end="")

    print(str2)

При чтении файла мы можем столкнуться с тем, что его кодировка не совпадает с ASCII. В этом случае мы явным образом можем указать кодировку с помощью параметра **encoding**:

filename = "hello.txt"

with open(filename, encoding="utf8") as file:

    text = file.read()

### Чтение и запись

Для открытия файла сразу для чтения и записи применяются флаги:

* **r+**: если файл не найден, то генерируется исключение FileNotFoundError
* **w+**: если файл не существует, то он автоматически создается. Если файл существует, то он перезаписывается

Например, запишем и считаем из файла данные:

with open("hello.txt", "w+") as file:

    file.write("Hello world\nHello work\n")  # сначала записываем данные

    file.seek(0)        # перемещаемся к первому байту в файле

    content = file.read()   # считываем данные

    print(content)

### Перемещение по файлу

Для перемещения по файлу после его открытия применяется метод **seek()**. В него передается индекс байта, к которому надо переместить указатель в файле. Например, в примере выше после записи указатель в файле автоматически устанавливается на конец файла. И считывания данных идет с позиции, на которой стоит указатель. И чтобы его сдвинуть в самое начало, применяется вызов

file.seek(0)

Аналогичным образом мы можем перемещаться по файлу на нужную позицию. Например, считаем данные, начиная с 5-го символа:

with open("hello.txt", "w+") as file:

    file.write("Hello world\n")  # сначала записываем данные

    file.seek(6)                # перемещаемся к шестому байту в файле

    content = file.read()       # считываем данные

    print(content)      # world

### Пример работы с файлами

Теперь напишем небольшой скрипт, в котором будет записывать введенный пользователем массив строк и считывать его обратно из файла на консоль:

# имя файла

FILENAME = "messages.txt"

# запись строки в файл

def write():

    message = input("Введите строку: ")

    with open(FILENAME, "a") as file:

        file.write(message + "\n")

# чтение файла файл

def read():

    with open(FILENAME, "r") as file:

        for message in file:

            print(message, end="")

    print() # перевод строки для разделения меню и вывода

while(True):

    selection = int(input("1.Запись в файл\t\t2.Чтение файла\t\t3.Выход\nВыберите действие: "))

    match selection:

        case 1: write()

        case 2: read()

        case 3: break

        case \_: print("Некорректный ввод")

print("Программа завершена")

Пример работы программы:

1.Запись в файл 2.Чтение файла 3.Выход

Выберите действие: 1

Введите строку: Hello World

1.Запись в файл 2.Чтение файла 3.Выход

Выберите действие: 1

Введите строку: Hello METANIT.COM

1.Запись в файл 2.Чтение файла 3.Выход

Выберите действие: 2

Hello World

Hello METANIT.COM

1.Запись в файл 2.Чтение файла 3.Выход

Выберите действие: 3

Программа завершена