**Работа с файлами в Python**

Прежде, чем работать с файлом, его надо открыть:

f = open('text.txt', 'r')

У функции open много параметров, нам пока важны 3 аргумента: первый, это имя файла. Путь к файлу может быть относительным или абсолютным. Второй аргумент, это режим, в котором мы будем открывать файл.

|  |  |
| --- | --- |
| 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 |

Режимы могут быть объединены, то есть, к примеру, 'rb' - чтение в двоичном режиме. По умолчанию режим равен 'rt'.

И последний аргумент, encoding, нужен только в текстовом режиме чтения файла. Этот аргумент задает кодировку.

**Чтение из файла**

Первый - метод read, читающий весь файл целиком, если был вызван без аргументов, и n символов, если был вызван с аргументом (целым числом n).

**>>>** f = open('text.txt')

**>>>** f.read(1)

'H'

**>>>** f.read()

'ello world!\nThe end.\n\n'

Ещё один способ сделать это - прочитать файл построчно, воспользовавшись циклом for:

**>>>** f = open('text.txt')

**>>> for** line **in** f:

**...**  line

**...**

'Hello world!\n'

'\n'

'The end.\n'

'\n'

**Запись в файл**

Теперь рассмотрим запись в файл. Попробуем записать в файл вот такой вот список (при помощи списочного выражения list comprehension):

**>>>** l = [str(i)+str(i-1) **for** i **in** range(20)]

**>>>** l

['0-1', '10', '21', '32', '43', '54', '65', '76', '87', '98', '109', '1110', '1211', '1312', '1413', '1514', '1615', '1716', '1817', '1918']

Откроем файл на запись:

**>>>** f = open('text.txt', 'w')

Запись в файл осуществляется с помощью метода write:

**>>> for** index **in** l:

**...**  f.write(index + '**\n**')

**...**

4

3

3

3

3

Метод write возвращает число записанных символов.

После окончания работы с файлом его **обязательно нужно закрыть** с помощью метода close:

**>>>** f.close()

Теперь попробуем воссоздать этот список из получившегося файла. Откроем файл на чтение, и прочитаем строки.

**>>>** f = open('text.txt', 'r')

**>>>** l = [line.strip() **for** line **in** f]

**>>>** l

['0-1', '10', '21', '32', '43', '54', '65', '76', '87', '98', '109', '1110', '1211', '1312', '1413', '1514', '1615', '1716', '1817', '1918']

**>>>** f.close()

**Работа с директориями в Python**

# Содержание папки

Модуль стандартной библиотеки os (от «operation system») предоставляет множество полезных функций для произведения системных вызовов. Одна из базовых функций этого модуля — os.listdir.

**>>>** **import** os

**>>>** file\_list **=** os.listdir() *# список файлов и папок в директории, где запущена программа*

**>>>** file\_list **=** os.listdir('.') *# синоним*

**>>>** file\_list **=** os.listdir('C:/Users') *# список имен файлов и папок в данной папке*

С точки зрения операционной системы нет разницы между файлом, папкой или другим подобным объектом, типа ссылки. Поэтому os.listdir() возвращает список как файлов, так и папок. Обратите внимание, что порядок элементов возвращаемого списка не регламентируется, если вам нужно их отсортировать не забудьте сделать это:

**>>>** **import** os

**>>>** unsorted\_file\_list **=** os.listdir()

**>>>** sortetd\_file\_list **=** sorted(unsorted\_file\_list)

## Создание папки

Для создания новой папки используйте os.mkdir(name). Эта функция выбросит ошибку, если по указанному пути уже существует файл или папка. Если вам нужно создать сразу несколько вложенных папок, то смотрите функцию [os.makedirs(name, exist\_ok=False)](https://docs.python.org/3/library/os.html" \l "os.makedirs).

## Удаление папки и файла

Удалить файл можно с помощью функции os.remove, а пустую папку с помощью функции os.rmdir.

**Дополнительные функции и методы**

**makedirs(path)**

- Работает аналогично функции mkdir(), но автоматически создает все необходимые промежуточные папки.

**removedirs(path)**

- Работает аналогично функции rmdir(), но автоматически удаляет все родительские пустые папки.

**rename(src, dst)**

- Переименовывает файл или папку с именем src в dst.

**readline()**

- Считывает из файла одну строку, включая символ перехода на новую строку («\n»).

**readlines()**

- Считывает из файла все строки и возвращает их в виде списка.

**writelines(lines)**

- Записывает строки из списка lines в файл. Символ перехода на новую строку между ними не добавляется.

**tell()**

- Возвращает текущее положение в файле в байтах от начала файла.

**seek(offset, whence)**

- Изменяет положение на offset. Характер определения положения зависит от аргумента whence (по умолчанию 0):

1) 0 – новое положение отсчитывается от начала файла;

2) 1 – от конца файла;

3) 2 – от текущего положения в файле;

**truncate(size)**

- Усекает файл до размера size.