Работа с файлами в Python

В данном материале рассмотрим встроенные средства Python для работы с файлами: открытие и закрытие файла, чтение данных из файла и запись данных в файл.

Прежде, чем работать с файлом, его надо открыть. Для этого используется встроенная функция **`open()`**, которая возвращает файловый объект, для того, чтобы получить доступ к чтению, записи или добавлению.

Вот пример ее использования:

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

У функции open много параметров, все эти параметры можно посмотреть в статье **Встроенные функции**. Нам же пока важны только три 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.

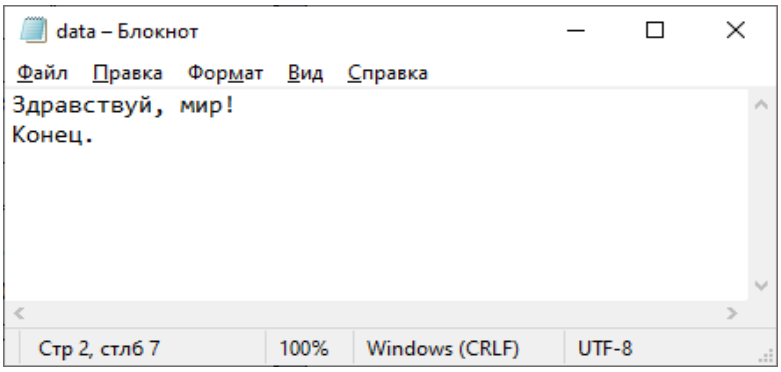
Режимы могут быть объединены. Например, `'rb'` - чтение в двоичном режиме. По умолчанию режим равен `'rt'`. Таким образом, если функция не содержит второй аргумент, файл расценивается как текстовый и открывается на чтение.

Попытка открыть на чтение несуществующий файл вызывает ошибку.

Еще один важный аргумент, **`encoding`**, нужен только в текстовом режиме чтения файла. Этот аргумент задает кодировку.

Чтение из файла

Создадим в обычном Блокноте (или любом другом текстовом редакторе) файл, содержащий всего две строчки:



Затем сохраним файл в той же папке, где будет находиться наша программа под именем **`data.txt`**.

Программа должна открыть файл с помощью функции **`open`**, а затем прочитать из него информацию. Для этого есть несколько способов, но мы рассмотрим только два из них.

Первый способ – это использование метода **`read`**, который читает весь файл целиком, если вызывается без аргументов, или читает **`n`** символов, если был вызван с аргументом (целым числом **`n`**).

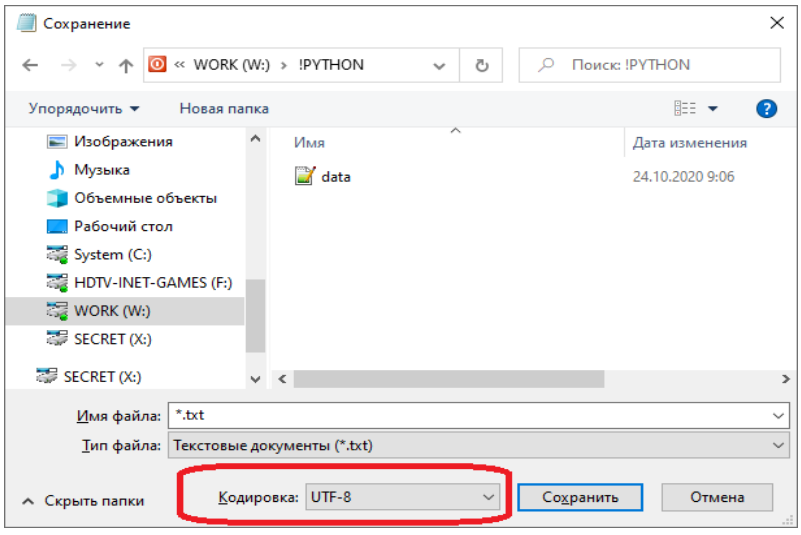
Создадим небольшую программу, содержащую всего три строчки:

```
f = open('data.txt')
print(f.read(5))
print(f.read())
```

Сохраним ее в той же папке, что и файл с данными, затем запустим. В результате получим “кракозябры”:

```
In [1]: runfile('W:/!PYTHON/probe-file-1.py')
Р—РґС
ЂР°РІС‚С,РІСґР№, РјРёСЂ!
РљРѕРЅРµС†.
```

Очевидно, что это проблема с кодировкой. Раньше кодировкой по умолчанию была **ANSI**, теперь это кодировка **UTF-8**:



Поэтому, чтобы избежать проблем с русскими буквами, мы добавим параметр **encoding='utf8'** в функцию **open**. Теперь наша программа выглядит так:

```
f = open('data.txt', encoding='utf8')
print(f.read(5))
print(f.read())
```

Результат будет выглядеть так:

```
In [2]: runfile('W:/!PYTHON/probe-file-1.py')
Здрав
ствуй, мир!
Конец.
```

Функция **print(f.read(5))** выводит первые 5 символов из файла, затем функция **print(f.read())** выводит все остальное вместе с переводом на новую строку.

Если в текущей папке файл с данными не обнаружен, то получим ошибку:

```
In [1]: runfile('W:/!PYTHON/probe-file-1.py')
Traceback (most recent call last):

  File "W:\!PYTHON\probe-file-1.py", line 7, in <module>
    f = open('data.txt', encoding='utf8')

FileNotFoundError: [Errno 2] No such file or directory: 'data.txt'
```

Второй способ - прочитать файл построчно, используя цикл **for**:

```
f = open('data.txt', encoding='utf8')
for a in f:
    print(a)
```

Результат:

```
In [5]: runfile('W:/!PYTHON/probe-file-1.py')
Здравствуй, мир!

Конец.
```

Запись в файл

Теперь рассмотрим запись в файл. Попробуем записать в файл вот такой список:

```
'1','2','3','4','5','10','11','12','13'
```

Программа для записи может, например, выглядеть так:

```
s = ['1','2','3','4','5','10','11','12','13']
f = open('out.txt', 'w')
for i in range(len(s)):
    f.write(s[i] + '\n')
f.close()
```

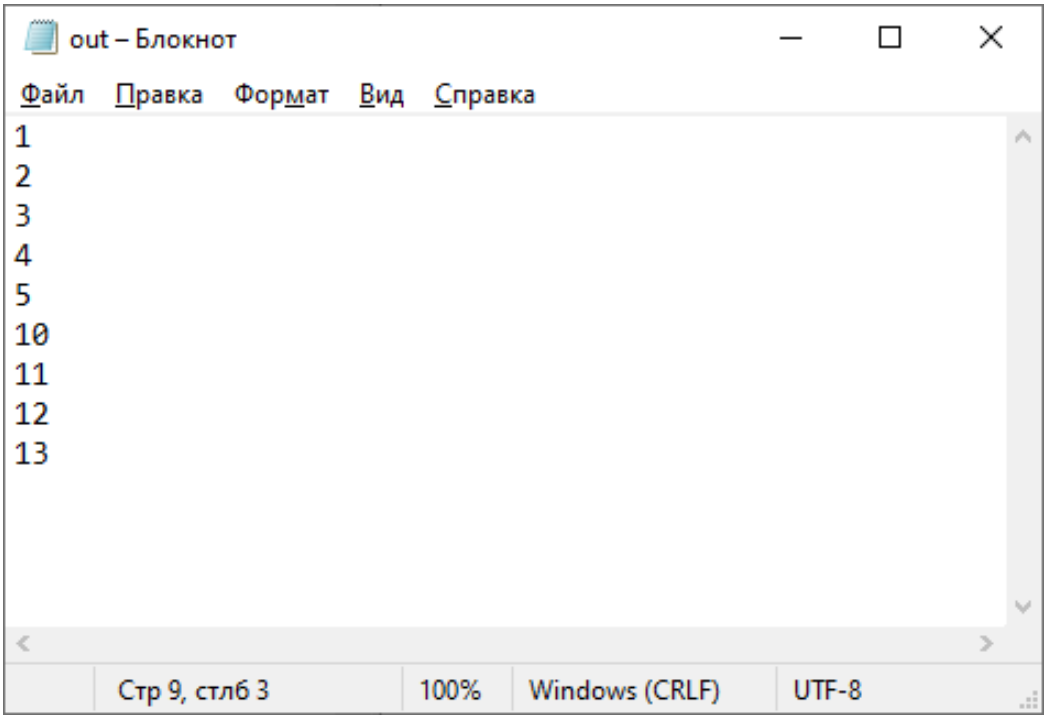
Инструкция **`f = open('out.txt', 'w')`** открывает файл **`out.txt`** на запись. **`f`** – это файловый объект. Метод **`write`** для объекта **`f`** осуществляет запись в файл: **`f.write()`**.

Экранированная последовательность **`\n`**, добавляемая в цикле после каждого элемента списка, осуществляет перевод строки, поэтому каждый элемент списка записывается с новой строки.

После окончания работы с файлом, его обязательно нужно закрыть. Для этого используется метод **`close`** для объекта **`f`**: **`f.close()`**.

После запуска вышеприведенной программы, в текущей папке будет создан новый файл **`out.txt`**, если его не было до этого. Если же файл существует, то он будет перезаписан.

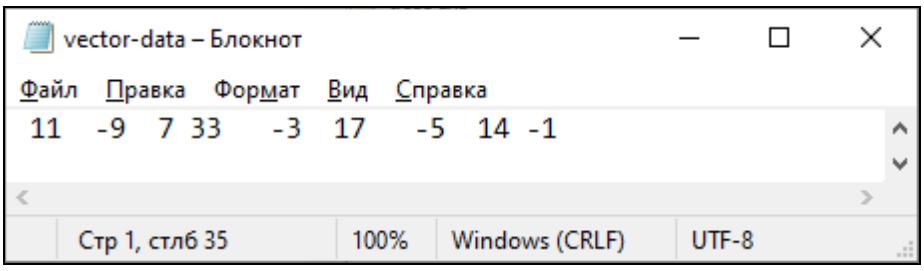
Полученный файл мы можем открыть в Блокноте:



Размер файла равен 31 байту (количество символов всех цифр равно 13, плюс после каждого числа добавлено по два символа перевода строки **`\n`**).

Ввод одномерного массива чисел из текстового файла

Создадим в обычном Блокноте (или в любом другом текстовом редакторе) файл, содержащий всего одну строку, в которой запишем числа, разделенные одним или несколькими пробелами:



Затем сохраним файл в той же папке, где будет находиться наша программа под именем **vector-data.txt**.

Наша программа должна открыть файл, а затем прочитать из него числовые данные и записать их в одномерный массив.

Пример программы приведен ниже:

```
fv = open('vector-data.txt')
vvod = fv.readline()
b = vvod.split()
for i in range(len(b)):
    b[i] = int(b[i])
```

Инструкция **fv = open('vector-data.txt')** открывает текстовый файл для чтения.

Инструкция **vvod = fv.readline()** благодаря методу **readline** записывает только одну строку из файла в строчную переменную **vvod**, которая теперь содержит все наши числа с пробелами.

Теперь нам надо разделить содержимое строки **vvod** на отдельные части. Для этого мы применяем метод **split()**.

Инструкция **b = vvod.split()** формирует список **b**, каждый элемент которого это отдельные числа из исходного файла в строковом виде. Эта инструкция как бы разрезала целую строку, вырезав из нее строковые элементы, используя пробелы в качестве разделителя.

Далее мы должны преобразовать элементы списка из строкового вида в числовой вид, также как мы это делаем при обычном вводе с клавиатуры.

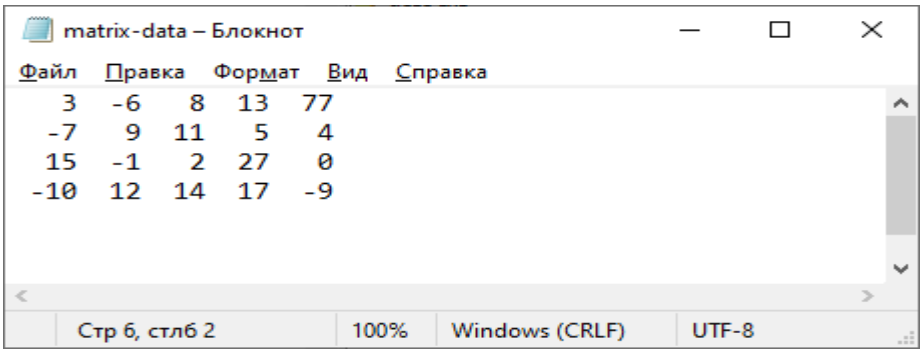
Для этого перебираем все элементы списка **b** при помощи цикла **for i in range(len(b)):** в котором записываем в каждый элемент списка преобразованное в соответствующий числовой тип значение: **b[i] = int(b[i])**. Если нужно считать список действительных чисел, то нужно заменить **int** на **float**.

В результате работы этой программы, мы получим одномерный массив **b**, который был введен не с клавиатуры, а из файла:

```
In [43]: b
Out[43]: [11, -9, 7, 33, -3, 17, -5, 14, -1]
```

Ввод двумерного массива чисел из текстового файла

Создадим в обычном Блокноте (или в любом другом текстовом редакторе) файл, содержащий элементы матрицы записанные построчно:



Затем сохраним файл в той же папке, где будет находиться наша программа под именем **matrix-data.txt**.

Наша программа должна открыть файл, а затем прочитать из него числовые данные и записать их в одномерный массив.

Пример программы приведен ниже:

```
fm = open('matrix-data.txt')
vvod = fm.readlines()
c = []
for i in range(len(vvod)):
    line = vvod[i]
    b = line.split()
    for j in range(len(b)):
        b[j] = int(b[j])
    if len(b) != 0:
        c.append(b)
```

Инструкция **fm = open('matrix-data.txt')** открывает текстовый файл для чтения.

Инструкция **vvod = fm.readlines()** благодаря методу **readlines** записывает весь файл в список **vvod**, каждый элемент этого списка это строка из исходного файла.

ВНИМАНИЕ! Эта инструкция не повторяет инструкцию из предыдущей программы для ввода одномерного массива! Метод readline читает ОДНУ СТРОКУ, а метод readlines читает ВЕСЬ ФАЙЛ.

Следующий фрагмент программы напоминает ввод двумерного массива с клавиатуры.

Создаем пустой список **c**, чтобы потом добавлять в него элементы: **c = []**.

Перебираем все строки (это количество элементов в списке vvod) в цикле **`for i in range(len(vvod)):`**.

Инструкция **`line = vvod[i]`** записывает **`i`**-ую строку из списка **`vvod`** в строчную переменную **`line`**.

Теперь нам надо разрезать **`line`** на отдельные строчные элементы, используя метод **`split()`** (точно так же, как мы это делали для строки **`vvod`** в предыдущей программе). Инструкция **`b = line.split()`** формирует список **`b`**, каждый элемент которого это отдельные числа из строки исходного файла в строковом виде.

Далее мы преобразовываем элементы списка **`b`** из строкового вида в числовой вид, также как мы это делаем при обычном вводе с клавиатуры.

Для этого перебираем все элементы списка **`b`** при помощи цикла **`for j in range(len(b)):`** в котором записываем в каждый элемент списка преобразованное в соответствующий числовой тип значение: **`b[j] = int(b[j])`**. Если нужно считать список действительных чисел, то нужно заменить **`int`** на **`float`**.

Полученный список **`b`** это числовые элементы строки матрицы, поэтому мы можем сразу добавить список **`b`** к двумерному списку **`c`**, используя инструкцию **`c.append(b)`**.

Однако будет нелишним делать это только при выполнении условия, которое проверяет что список **`b`** не пустой: **`if len(b) != 0:`**. Таким образом, мы отсекаем пустые строки из исходного файла.

В результате работы этой программы, мы получим двумерный массив **`c`**, который был введен не с клавиатуры, а из файла:

```
In [49]: c
Out[49]:
[[3, -6, 8, 13, 77],
 [-7, 9, 11, 5, 4],
 [15, -1, 2, 27, 0],
 [-10, 12, 14, 17, -9]]
```