Чтение файлов в формате CSV в Python

**ОПУБЛИКОВАНО**[**04.07.2021**](https://pythonist.ru/chtenie-formata-csv-v-python/)[**МОДУЛИ**](https://pythonist.ru/category/modules/)

*Перевод*[*статьи*](https://linuxhint.com/python-read-csv-tutorial/)*«Python Read CSV Tutorial».*

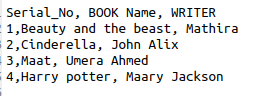
CSV расшифровывается как «Comma Separated Values», т.е. «значения, разделенные запятыми». Это самый простой способ хранения данных в виде обычного текста. Данные представлены в табличной форме, где каждая строка является строкой записи таблицы. Сегодня мы поговорим про чтение формата CSV в Python. Расскажем о различных функциях для работы с CSV-файлами, а также о том, как создать свой файл CSV.

**Чтение CSV-файла при помощи csv.reader(). Метод 1**

**Пример 1. Использование запятой в качестве разделителя**

Допустим, у нас есть файл с именем sample1, содержащий некоторые данные. Такой файл можно создать с помощью любого текстового редактора или путем передачи значений в какую-нибудь программу для записи файла CSV. Как это делается, мы расскажем чуть позже.

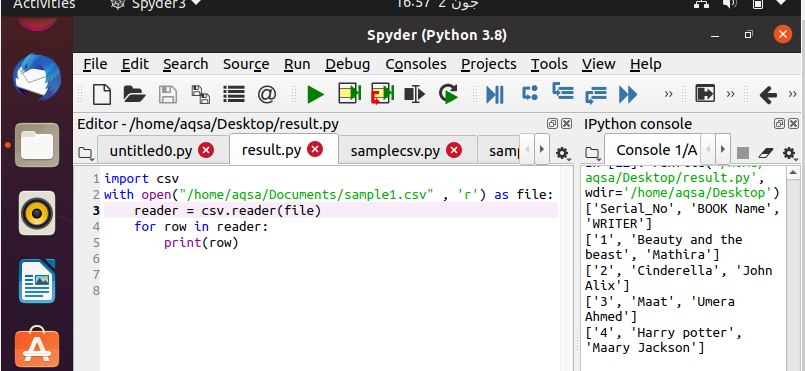
Текст в этом файле разделен запятыми. Наши данные — сведения о различных книгах (порядковый номер, название, имя автора).



Переходим к коду. Чтобы прочитать файл CSV, нам нужен объект reader для выполнения функции чтения. Первым делом импортируем модуль csv (встроенный модуль Python). Далее укажем имя файла, который нужно открыть (или путь к нему). Затем инициализируем объект reader. Он будет перебираться в цикле for.

reader = csv.reader(файл)

Данные из указанного файла выводятся построчно.



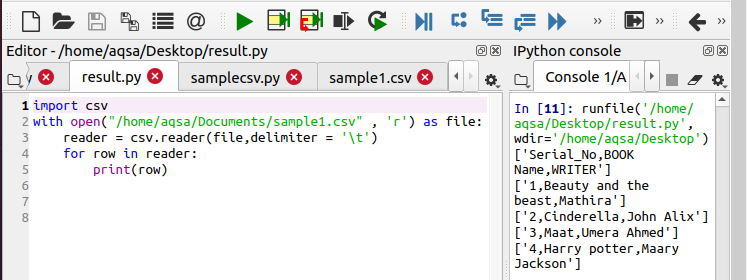
Запустим наш код. Можем увидеть, что наши данные автоматически преобразовались в [списки](https://pythonist.ru/spiski-v-python-polnoe-rukovodstvo-dlya-nachinayushhih/), состоящие из элементов данных каждой строки.

**Пример 2. Использование табуляции в качестве разделителя**

В первом примере текст разделяется запятой. Однако мы можем сделать наш код более настраиваемым, добавив различные функции.

Возьмём код из предыдущего примера и сделаем лишь одно изменение: напрямую укажем разделитель (delimiter). В предыдущем примере не было никакой необходимости его указывать, потому что запятая — разделитель по умолчанию.

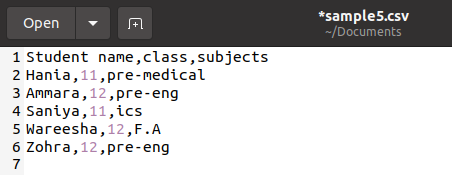
reader = csv.reader(file, delimiter = ‘\t’)



Как видите, результат немного изменился. Теперь все элементы строки являются одним элементом списка. Так произошло потому, что в этот раз мы указали в качестве разделителя не запятую, а \t.

**Чтение CSV-файла при помощи csv.reader(). Метод 2**

Теперь давайте обсудим альтернативный метод чтения файлов CSV. Предположим, у нас есть файл sample5, сохраненный с расширением .csv. В файле сохранены данные об учениках, их именах, классе и предметах.

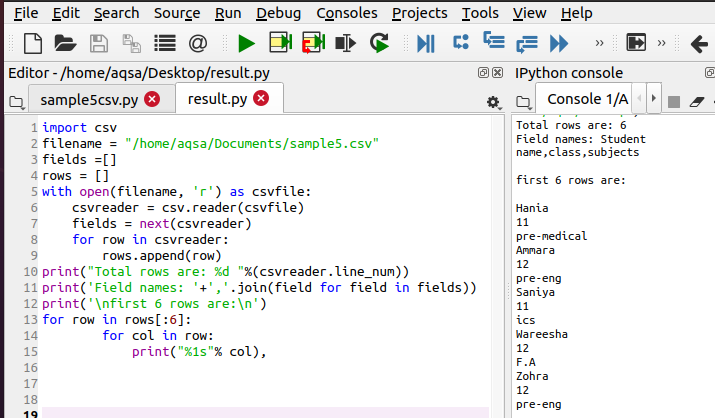


Теперь перейдем к коду. Первый шаг такой же, как и в прошлый раз – импорт модуля. Затем укажем путь или имя файла, который необходимо открыть и использовать.

Этот код является примером одновременного чтения и изменения данных. Мы инициализируем два массива для будущего использования в этом коде. Затем открываем файл с помощью функции open(). Далее инициализируем объект, как мы это делали в приведенных выше примерах. Здесь снова используется цикл for.

Следующая функция сохраняет текущее значение строк и пересылает объект для следующей итерации:

fields = next(csvreader)



Все строки добавляются к списку с именем rows.

rows.append(row)

Если мы хотим увидеть общее количество строк, вызовем следующую функцию вывода.

print(“total rows are: %d “%(csvreader.line\_num))

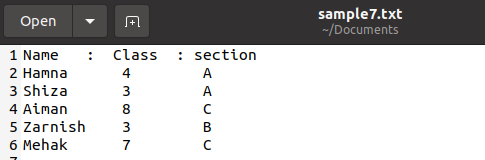
Чтобы вывести заголовок столбца или имя поля, используем функцию, в которой к тексту (с помощью метода .join()) присоединяются все заголовки.

После выполнения вы можете увидеть результат, в котором каждая строка напечатана с полным описанием и текстом, который мы добавили с помощью кода во время выполнения.

**Использование DictReader**

Эта функция используется для вывода данных из текстового файла в виде [словаря](https://pythonist.ru/kak-ustroeny-slovari-v-python/).

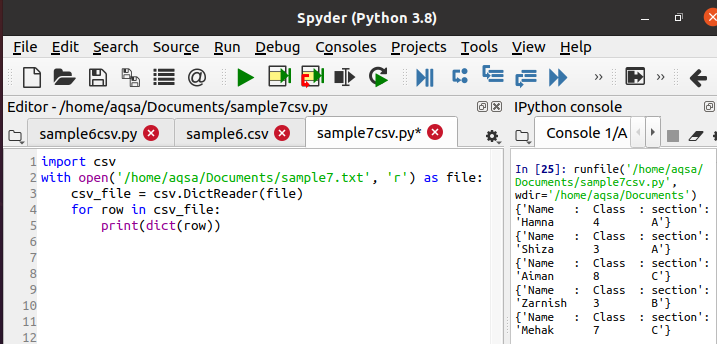
У нас есть файл sample7.txt со следующими данными об учениках.



**Примечание.**Не обязательно сохранять файл с расширением .csv, можно и в других форматах. Главное, чтобы в самом файле использовался простой текст и данные оставались в первозданном виде.

Теперь воспользуемся приведенным ниже кодом, чтобы прочитать данные и вывести их в формате словаря. Вся методика одинакова, только вместо reader используется DictReader.

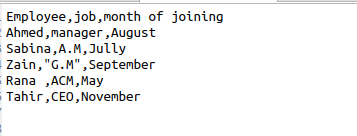
csv\_file = csv.DictReader(file)



Во время выполнения можно видеть, что данные печатаются в виде словаря. Данная функция преобразует каждую строку в словарь.

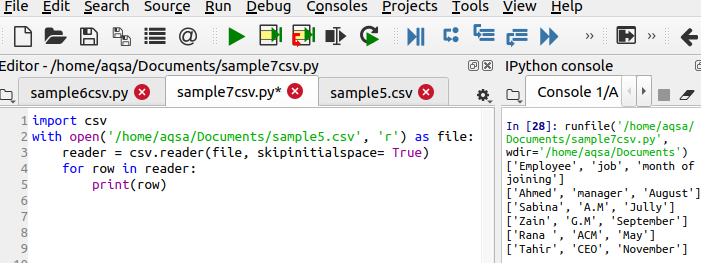
**Пробелы в файлах CSV**

Каждый раз при использовании csv.reader() мы автоматически получаем пробелы в выводе. Чтобы удалить эти лишние пробелы из вывода, нам нужно использовать функцию reader(), указав при этом специальный параметр. Предположим, файл содержит следующие данные, касающиеся информации о сотруднике.



reader = csv.reader(file, skipinitialspace = **True**)

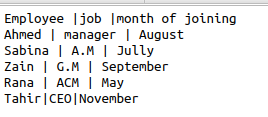
Укажем значение skipinitialspace, равное True. Благодаря этому неиспользуемое свободное пространство удаляется из вывода.



**Модуль CSV и диалекты**

Если мы начнем работать, используя одни и те же файлы csv с форматами функций в коде, это сделает код очень некрасивым и непоследовательным. Модуль CSV помогает использовать метод диалектов в качестве опции для удаления избыточности данных.

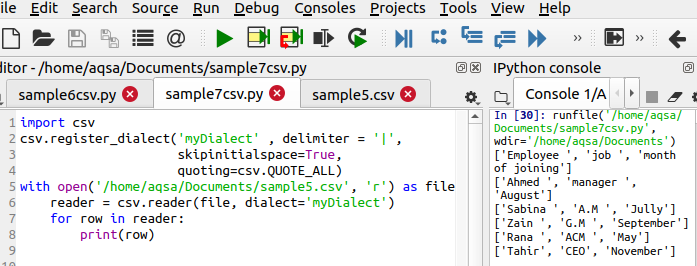
Рассмотрим в качестве примера те же исходные данные, но на этот раз разделенные символом |. Мы хотим удалить этот символ, пропустить лишний пробел и использовать одинарные кавычки между соответствующими данными.



Используя следующую строчку кода, мы получим желаемый результат

csv.register\_dialect(‘myDialect’, delimiter =’|’, skipinitialspace=**True**, quoting= csv.QUOATE\_ALL)

Эта строка отличается, поскольку она определяет три основные функции, которые необходимо выполнить. Из выходных данных вы можете видеть, что символ | удаляется, и также добавляются одинарные кавычки.



**Написание своего CSV-файла**

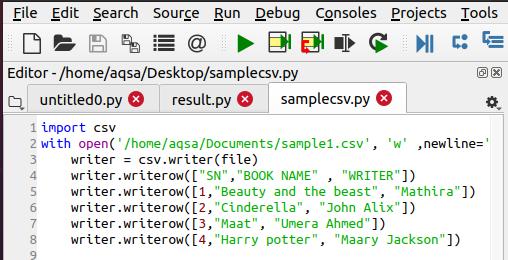
Чтобы открыть файл csv, нам нужно его где-то взять. Если его нет, можно создать такой файл с помощью следующей функции.

Для начала импортируем модуль csv. Затем дадим имя файлу, который хотим создать. Для добавления данных мы будем использовать следующий код:

writer = csv.writer(file)

writer.writerow(……)

Данные вводятся в файл построчно, поэтому следует использовать именно этот оператор.



**Заключение**

В этой статье мы разобрали тему чтения формата CSV в Python. Мы показали, как вывести данные с помощью различных методов, в виде словарей или списков. Также мы рассказали, как удалить лишние пробелы и специальные символы из данных.