Формат TSV

TSV

TSV (англ. tab separated values — «значения, разделенные табуляцией») — текстовый формат для представления таблиц баз данных. Каждая запись в таблице — строка текстового файла. Каждое поле записи отделяется от других символом табуляции, а точнее — горизонтальной табуляции.

TSV — форма более общего формата **DSV** («значения, разграниченные разделителем», от англ. delimiter separated values).

Программы для работы с электронными таблицами (MS Excel, LibreOffice Calc) поддерживают импорт из такого формата.

Импорт из текстового файла в Excel

Посмотрим, как работать с таким форматом в Python. Тут нет ничего сложного, поскольку у нас есть мощная функциональность по работе со строками, в частности, метод split.

```
data = open('ikea.txt', encoding='utf-8').readlines()
for row in data[:10]:
    print(row.rstrip().split('\t'))
```

```
['keywords', 'price', 'product_name']
['МОРУМ, Ковёр, безворсовый', '6999', 'МОРУМ']
['МОРУМ, Ковёр, безворсовый', '6999', 'МОРУМ']
['ИДБИ, Придверный коврик', '649', 'ИДБИ']
['ХОДДЕ, Ковёр, безворсовый', '1399', 'ХОДДЕ']
['ОПЛЕВ, Придверный коврик', '599', 'ОПЛЕВ']
['ОПЛЕВ, Придверный коврик', '599', 'ОПЛЕВ']
['ЮНКЭН, Брикеты', '89', 'ЮНКЭН']
['БУНСЁ, Детское садовое кресло', '1199', 'БУНСЁ']
['ИКЕА ПС ВОГЭ, Садовое лёгкое кресло', '1999', 'ИКЕА ПС ВОГЭ']
```

Вспомним списочные выражения и сразу сделаем «двумерный массив», а потом обратимся к цене пятого по счету товара:

```
table = [r.rstrip().split('\t') for r in data]
print(table[5][1])
```

```
599
```

Мы можем также отсортировать элементы по цене и напечатать 10 самых дешевых товаров:

```
table = table[1:]
table.sort(key=lambda x: int(x[1]))
for r in table[:10]:
    print(r)
```

```
['СМОРИСКА, Стопка', '6', 'СМОРИСКА']
['СМОРИСКА, Стакан', '12', 'СМОРИСКА']
['ДИСТАНС, Контейнер', '12', 'ДИСТАНС']
['ДИСТАНС, Контейнер', '12', 'ДИСТАНС']
['ОППЕН, Миска', '25', 'ОППЕН']
['ДАРРОКА, Стакан', '25', 'ДАРРОКА']
['АНТАГЕН, Щетка для мытья посуды', '25', 'АНТАГЕН']
['ВАНКИВА, Рама', '25', 'ВАНКИВА']
['ХОППЛЁС, Доска разделочная', '27', 'ХОППЛЁС']
['ДАРРОКА, Стакан д/виски', '29', 'ДАРРОКА']
```

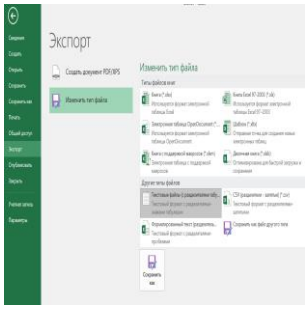
Формат CSV

Одним из самых распространенных форматов **DSV** стал **CSV** — формат с разделителем полей-запятой (англ. comma separated values). Наш файл будет выглядеть в нем вот так:

```
> keywords,price,product_name

>"МОРУМ, Ковёр, безворсовый",6999,МОРУМ

>"МОРУМ, Ковёр, безворсовый",6999,МОРУМ

>"ИДБИ, Придверный коврик",649,ИДБИ

>"ХОДДЕ, Ковёр, безворсовый",1399,ХОДДЕ

>"ОПЛЕВ, Придверный коврик",599,ОПЛЕВ

>"ОПЛЕВ, Придверный коврик",599,ОПЛЕВ

>"ЮНКЭН, Брикеты",89,ЮНКЭН

>"БУНСЁ, Детское садовое кресло",1199,БУНСЁ

>"ИКЕА ПС ВОГЭ, Садовое лёгкое кресло",1999,ИКЕА ПС ВОГЭ
```

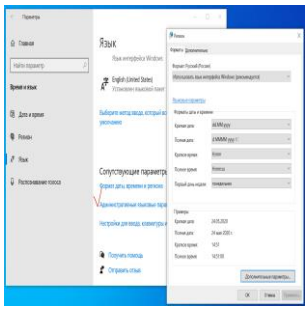
Для всех форматов DSV **проблемой** является символ-разделитель полей в данных. В этом случае вводят так называемый **разделитель текста**, в качестве которого выступают двойные кавычки, а если в поле встречается сам разделитель текста, то его удваивают. Например, **ООО "Светлана"** при записи в файл превращается в **"ООО ""Светлана"""**.

Вот почему в приведенном фрагменте CSV-файла первое поле в кавычках — внутри него есть запятые.

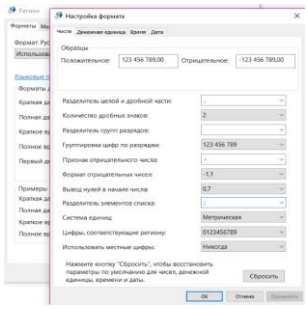


Нужно отметить, что в CSV могут быть другие разделители, например, точка с запятой. Очень часто это регулируется настройками операционной системы (параметр «Разделитель элементов списка» в ОС Windows):

Параметры — Время и язык — Язык:



Дополнительные параметры:



Понятие о модулях в Python
Вы уже знаете, что код можно переиспользовать в программе, для этого его можно выделить в виде функции, дать ей имя, а затем вызывать по мере надобности. Также функции можно вынести в отдельный файл (модуль), который подключается к основной программе, в которой вы хотите его использовать, с помощью импорта.

Разберем пример.

Создайте в папке проекта файл functions.py с функцией power_x(x, pow):

```
def power_x(x, pow):
    return x ** pow
```

Теперь создайте файл main.py в той же папке, в котором мы импортируем файл с функцией, и вызовем ее:

```
import functions

print(functions.power_x(2, 11))  # 2048
```

Мы импортировали все содержимое файла functions.py, а можно было импортировать только необходимую нам функцию:

```
from functions import power_x

print(power_x(2, 11))
```

Так лучше поступать, если в файле много функций, а нам нужна только одна.

Библиотеки в Python — это такие готовые модули, в которых есть функции (и не только), которые написаны другими программистами и которыми мы можем пользоваться.

Рассмотрим, как пользоваться библиотекой, предназначенной для работы с csv-файлами.

Библиотека CSV

Несмотря на то, что DSV-форматы просты, отсутствие четких стандартов в выборе разделителей и экранировании символов привели к тому, что с ними лучше работать при помощи специализированных библиотек, а не в стиле «использования функции» split.

Для работы с такими форматами в Python есть модуль csv.

В модуле есть два основных объекта: reader и writer, созданные, чтобы читать и создавать csv-файлы соответственно.

Приведем пример использования **читателя** с почти полным набором значений, указав:

- Кодировку файла
- Символ-разделитель
- Разделитель текста

Объект reader дает доступ к построчному итератору полностью аналогично работе с файлом или списком.

Общности ради в следующем примере мы покажем, что разделителем может быть любой символ.

```
import csv
with open('files/ikea.csv', encoding="utf8") as csvfile:
    reader = csv.reader(csvfile, delimiter=';', quotechar='"')
    for index, row in enumerate(reader):
        if index > 10:
            break
        print(row)
['keywords', 'price', 'product_name']
['МОРУМ, Ковёр, безворсовый', '6999', 'МОРУМ']
['МОРУМ, Ковёр, безворсовый', '6999', 'МОРУМ']
['ИДБИ, Придверный коврик', '649', 'ИДБИ']
['ХОДДЕ, Ковёр, безворсовый', '1399', 'ХОДДЕ']
['ОПЛЕВ, Придверный коврик', '599', 'ОПЛЕВ']
['ОПЛЕВ, Придверный коврик', '599', 'ОПЛЕВ']
['ЮНКЭН, Брикеты', '89', 'ЮНКЭН']
['БУНСЁ, Детское садовое кресло', '1199', 'БУНСЁ']
['ИКЕА ПС ВОГЭ, Садовое лёгкое кресло', '1999', 'ИКЕА ПС ВОГЭ']
['КУНГСХОЛЬМЕН, Садовый табурет', '5500', 'КУНГСХОЛЬМЕН']
```

Давайте разберем построчно, что происходит в этом коде.

Мы пользуемся `with`, чтобы просто открыть наш файл с кодировкой UTF-8, а потом создаем объект `reader`, говоря ему про символы-разделители полей и строк.

Объект `reader` может служить **итератором** (и использоваться в цикле `for`) по строкам, каждая из которых представляет собой список. Кроме того, мы используем `enumerate`, чтобы посчитать строки.

Отметим, что исходный файл содержит подписи полей в первой строке, что будет снова нам мешать (например, при сортировке строк). Для корректной работы мы должны были бы исключить первую строку из обработки.

С другой стороны, названия полей тоже нужно сохранить. Вспомним, что `reader` – итератор, и получим заголовки методом `next`:

```
import csv

with open('files/ikea.csv', encoding="utf8") as csv_file:
    reader = csv.reader(csv_file)
    headers = next(reader)
    print(headers)
    print(*reader, sep='\n')
```

В таком случае мы сначала получили первый элемент итератора – список заголовков, а затем вывели все остальное, что в нем осталось.

Но в модуле csv есть специальный объект **DictReader**, который поддерживает создание объекта-словаря на основе подписей к полям.

DictReader не просто словарь, а словарь, который отслеживает порядок ключей после их добавления, — **OrderedDict**, что будет дальше видно в примере. Дополнительно почитать про **OrderedDict** можно тут.

Теперь мы можем обращаться к полям не по индексу, а по **названию**, что делает программу еще более понятной.

Найдем топ-10 самых дорогих товаров (как вы думаете, какая запись более понятна: int(x['price']) или int(x[1])?):

```
with open('files/ikea.csv', encoding="utf8") as csvfile:
    reader = csv.DictReader(csvfile, delimiter=';', quotechar='"')
    expensive = sorted(reader, key=lambda x: int(x['price']),
                       reverse=True)

for record in expensive[:10]:
    print(record)
OrderedDict([('keywords', 'ГРИЛЬЕРА, Плита'), ('price', '99999'),
('product_name', 'ГРИЛЬЕРА')])
OrderedDict([('keywords', 'ГРИЛЬЕРА, Плита'), ('price', '99999'),
('product_name', 'ГРИЛЬЕРА')])
OrderedDict([('keywords', 'КИВИК, Диван-кровать 3-местный'),
('price', '79999'), ('product_name', 'КИВИК')])
OrderedDict([('keywords', 'КИВИК, Диван-кровать 3-местный'),
('price', '79999'), ('product_name', 'КИВИК')])
OrderedDict([('keywords', 'СТОКГОЛЬМ, Диван 3-местный'), ('price',
'69999'), ('product_name', 'СТОКГОЛЬМ')])
OrderedDict([('keywords', 'ИСАНДЕ, Встраив холодильник/морозильник
A++'), ('price', '59999'), ('product_name', 'ИСАНДЕ')])
OrderedDict([('keywords', 'КУЛИНАРИСК, Комбинир СВЧ с горячим
обдувом'), ('price', '54999'), ('product_name', 'КУЛИНАРИСК')])
OrderedDict([('keywords', 'ХОГКЛАССИГ, Индукц варочн панель'),
('price', '49999'), ('product_name', 'ХОГКЛАССИГ')])
OrderedDict([('keywords', 'ГРЭНСЛЁС, Комбинир СВЧ с горячим
обдувом'), ('price', '49999'), ('product_name', 'ГРЭНСЛЁС')])
OrderedDict([('keywords', 'КУЛИНАРИСК, Духовка/пиролитическая
самоочистка'), ('price', '49999'), ('product_name', 'КУЛИНАРИСК')])
```

Мы привели цены к типу int, потому что строки сравниваются в лексикографическом порядке (по алфавиту). Например:

```
print('11' > '100')
Trueprint(11 > 100)
False
```

Использование объекта для записи (writer) аналогично «читателю» (reader):

```
with open('files/квадраты.csv', 'w', newline='') as csvfile:
    writer = csv.writer(
        csvfile, delimiter=';', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
    for i in range(10):
        writer.writerow([i, i ** 2, f"Квадрат числа {i} равен {i **
2}"])
```

В этом случае использовался опциональный параметр функции open newline. Он отвечает за переводы строк при чтении или записи в текстовый файл. По умолчанию имеет значение None, в этом случае все разделители строк преобразуются в \n. Если в файле оказывается лишний перевод строки, то следует использовать этот параметр в режиме newline='', тогда \n будет преобразован в пустую строку.

Выполните этот код и посмотрите, что получилось.

Записывать в csv-файл можно и с помощью DictWriter, аналогичного DictReader. Но нужно ему указать, какие заголовки должны быть в файле и какие значения им соответствуют у каждой записи. Для этого сначала подготовим список словарей:

```
import csv

data = [{
    'lastname': 'Иванов',
    'firstname': 'Пётр',
    'class_number': 9,
    'class_letter': 'А'
}, {
    'lastname': 'Кузнецов',
    'firstname': 'Алексей',
    'class_number': 9,
    'class_letter': 'В'
}, {
    'lastname': 'Меньшова',
    'firstname': 'Алиса',
    'class_number': 9,
    'class_letter': 'А'
}, {
    'lastname': 'Иванова',
    'firstname': 'Татьяна',
    'class_number': 9,
    'class_letter': 'Б'
```

```
}]

with open('dictwriter.csv', 'w', newline='') as f:
    writer = csv.DictWriter(
        f, fieldnames=list(data[0].keys()), delimiter=';',
        quoting=csv.QUOTE_NONNUMERIC)
    writer.writeheader()
    for d in data:
        writer.writerow(d)
```

Результат:

```
"lastname";"firstname";"class_number";"class_letter"
"Иванов";"Пётр";9;"А"
"Кузнецов";"Алексей";9;"В"
"Меньшова";"Алиса";9;"А"
"Иванова";"Татьяна";9;"Б"
```