**Лекция 1**

**Импорт данных в Python из файлов CSV**

Программисты часто сталкиваются с задачей обработки больших объемов структурированных данных. Python имеет встроенную библиотеку CSV, с помощью которой программист может работать со специальными CSV файлами. Это своего рода электронные таблицы.

**Файл CSV** – это особый вид файла, который позволяет структурировать большие объемы данных.

По сути, он является обычным текстовым файлом, однако каждый новый элемент отделен от предыдущего запятой или другим разделителем. Обычно каждая запись начинается с новой строки. Данные CSV можно легко экспортировать в электронные таблицы или базы данных. Программист может расширять CSV файл, добавляя новые строки.

Пример CSV файла, где в качестве разделителя используется запятая:

Имя,Профессия,Год рождения

Виктор,Токарь, 1995

Сергей,Сварщик,1983

Как видно из примера, в первой строке обычно указывается, какая информация будет находиться в каждом столбце. Кроме того, после последнего элемента строки запятая не ставиться, интерпретатор определяет конец строки по символу переноса.

Вместо запятой можно использовать любой другой разделитель, поэтому при чтении CSV файла нужно заранее знать, какой символ используется.

Важно помнить, что CSV – это обычный текстовый файл, который не поддерживает символы в кодировках, отличающихся от ASCII или Unicode.

**Библиотека CSV**

Эта основная библиотека для работы с CSV файлами в Python.

Библиотека csv является встроенной, поэтому её не нужно скачивать, достаточно использовать обычный импорт:

import csv

**Чтение из файлов (парсинг)**

Для того чтобы прочитать данные из файла, программист должен создать объект reader:

reader\_object = csv.reader(file, delimiter = ",")

reader имеет метод next(), то есть является итерируемым объектом, поэтому чтение из файла происходит следующим образом:

import csv

with open("classmates.csv", encoding='utf-8') as r\_file:

# Создаем объект reader, указываем символ-разделитель ","

file\_reader = csv.reader(r\_file, delimiter = ",")

# Счетчик для подсчета количества строк и вывода заголовков столбцов

count = 0

# Считывание данных из CSV файла

for row in file\_reader:

if count == 0:

# Вывод строки, содержащей заголовки для столбцов

print(f'Файл содержит столбцы: {", ".join(row)}')

else:

# Вывод строк

print(f' {row[0]} - {row[1]} и он родился в {row[2]} году.')

count += 1

print(f'Всего в файле {count} строк.')

Предположим, что у нас есть CSV файл, который содержит следующую информацию:

Имя,Успеваемость,Год рождения

Саша,отличник,200

Маша,хорошистка,1999

Петя,троечник,2000

Тогда, если открыть этот файл в нашей программе, то будут получены следующие результаты:

Файл содержит столбцы: Имя, Успеваемость, Год рождения

Саша - отличник и он родился в 200 году.

Маша - хорошистка и он родился в 1999 году.

Петя - троечник и он родился в 2000 году.

Всего в файле 4 строк.

Использование конструкции with…as позволяет программисту быть уверенным, что файл будет закрыт, даже если при выполнении кода произойдет какая-то ошибка.

Обратите внимание, что при открытии нужно указать правильную кодировку, в которой сохранены данные. В данном случае encoding=’utf-8′. Если не указывать, то будет использоваться кодировка по умолчанию. Для Windows это cp1251.

Библиотека CSV позволяет работать с файлами, как со **словарями**, для этого нужно создать объект DictReader. Обращаться к элементам можно по имени столбцов, а не с помощью индексов. Для того, чтобы исходная программа делала аналогичный вывод, её следует изменить следующим образом:

import csv

with open("classmates.csv", encoding='utf-8') as r\_file:

# Создаем объект DictReader, указываем символ-разделитель ","

file\_reader = csv.DictReader(r\_file, delimiter = ",")

# Счетчик для подсчета количества строк и вывода заголовков столбцов

count = 0

# Считывание данных из CSV файла

for row in file\_reader:

if count == 0:

# Вывод строки, содержащей заголовки для столбцов

print(f'Файл содержит столбцы: {", ".join(row)}')

# Вывод строк

print(f' {row["Имя"]} - {row["Успеваемость"]}', end='')

print(f' и он родился в {row["Год рождения"]} году.')

count += 1

print(f'Всего в файле {count + 1} строк.')

Обращаться к элементам по названию столбца более удобно, кроме того, это упрощает понимание кода.

Обратите внимание, что в цикл for при первой итерации будет записан в row не шапка таблицы, а первая её строка. Поэтому при выводе количества строк переменную count увеличили на 1.

**Дополнительные параметры объекта DictReader**

DictReader имеет параметры:

**dialect** — Набор параметров для форматирования информации. Подробнее про них ниже.

**line\_num** — Устанавливает количество строк, которое может быть прочитано.

**fieldnames** — Определяет заголовки для столбцов, если не определить атрибут, то в него запишутся элементы из первой прочитанной строки файла. Заголовки нужны для того, чтобы легко было понять, какая информация содержится или должна содержаться в столбце.

Например, если бы в classmates.csv не было бы первой строки с заголовками, то можно было бы его открыть следующим образом:

fieldnames = ['Имя', 'Успеваемость', 'Год рождения']

file\_reader = csv.DictReader(r\_file, fieldnames = fieldnames)

Также можно использовать метод next() для получения следующей строки. Этот метод делает объект reader итерируемым. То есть он вызывается при каждой итерации и возвращает следующую строку. Этот метод и используется при каждой итерации в цикле for для получения очередной строки.

**Запись в файлы**

Для записи информации в CSV файл необходимо создать объект writer:

file\_writer = csv.writer(w\_file, delimiter = "\t")

Для записи в файл данных используется метод writerow(), который имеет следующий синтаксис:

writecol("Имя", "Фамилия", "Отчество")

Код программы для записи в CSV файл выглядит так:

import csv

with open("classmates.csv", mode="w", encoding='utf-8') as w\_file:

file\_writer = csv.writer(w\_file, delimiter = ",", lineterminator="\r")

file\_writer.writerow(["Имя", "Класс", "Возраст"])

file\_writer.writerow(["Женя", "3", "10"])

file\_writer.writerow(["Саша", "5", "12"])

file\_writer.writerow(["Маша", "11", "18"])

Обратите внимание, что при записи использовался, lineterminator="\r". Это разделитель между строками таблицы, по умолчанию он "\r\n".

После выполнения программы в файле CSV будет следующий текст:

Имя,Класс,Возраст

Женя,3,10

Саша,5,12

Маша,11,18

В качестве параметра метод writerow() принимает список, элементы которого будут записаны в строку через символ-разделитель.

Запись в файл также может быть осуществлена с помощью объекта DictWriter. Важно помнить, что он требует явного указания параметра fieldnames. В качестве аргумента метода writerow используется словарь.

Код программы выглядит так:

import csv

with open("classmates.csv", mode="w", encoding='utf-8') as w\_file:

names = ["Имя", "Возраст"]

file\_writer = csv.DictWriter(w\_file, delimiter = ",",

lineterminator="\r", fieldnames=names)

file\_writer.writeheader()

file\_writer.writerow({"Имя": "Саша", "Возраст": "6"})

file\_writer.writerow({"Имя": "Маша", "Возраст": "15"})

file\_writer.writerow({"Имя": "Вова", "Возраст": "14"})

Вывод в файл будет следующим:

Имя,Возраст

Саша,6

Маша,15

Вова,14

**Дополнительные параметры DictWriter**

Объект writer также имеет атрибут dialect, который определяет, как будут форматироваться данные при записи в файл, про него будет описано ниже.

Кроме того, writer имеет методы:

**writerows(rows)** — Записывает все элементы строк.

**writeheader()** — Выводит заголовки для столбцов. Заголовки должны быть переданы объекту writer в виде списка, как атрибут fieldnames.

**writeheader** был использован в предыдущем примере. Рассмотрим применение writerows:

file\_writer.writerows([{"Имя": "Саша", "Возраст": "6"},

{"Имя": "Маша", "Возраст": "15"},

{"Имя": "Вова", "Возраст": "14"}])

**Диалекты**

Чтобы каждый раз не указывать формат входных и выходных данных, определенные параметры форматирования сгруппированы в диалекты (dialect). При создании объекта reader или writer программист может указать нужный ему диалект, кроме того, некоторые параметры диалекта можно переопределить вручную, также указав их при создании объекта.

Для создания диалекта используется команда:

register\_dialect("имя", delimiter = "\t", ...)

Класс Dialect позволяет определить следующие атрибуты форматирования:

| **Атрибут** | **Значение** |
| --- | --- |
| delimiter | Устанавливает символ, с помощью которого разделяются элементы в файле. По умолчанию используется запятая. |
| doublequote | Если True, то символ quotechar удваивается, если False, то к символу qutechar добавляется ecsapechar в качестве префикса. |
| escapechar | Строка из одного символа, которая используется для экранирования символа-разделителя. |
| lineterminator | Определяет разделитель для строк, по умолчанию используется «\r\n» |
| quotechar | Определяет символ, который используется для окружения символа-разделителя. По умолчанию используются двойные кавычки, то есть quotechar = ‘ » ‘. |
| quoting | Определяет символ, который используется для экранирования символа разделителя (если не используются кавычки). |
| skipinitialspace | Если установить значение этого параметра в True, то все пробелы после символа-разделителя будут игнорироваться. |
| strict | Если установить в True, то при неправильном вводе CSV будет возбуждаться исключение Error. |

Пример использования:

import csv

csv.register\_dialect('my\_dialect', delimiter=':', lineterminator="\r")

with open("classmates.csv", mode="w", encoding='utf-8') as w\_file:

file\_writer = csv.writer(w\_file, 'my\_dialect')

file\_writer.writerow(["Имя", "Класс", "Возраст"])

file\_writer.writerow(["Женя", "3", "10"])

file\_writer.writerow(["Саша", "5", "12"])

file\_writer.writerow(["Маша", "11", "18"])

В результате получим:

Имя:Класс:Возраст

Женя:3:10

Саша:5:12

Маша:11:18

l())