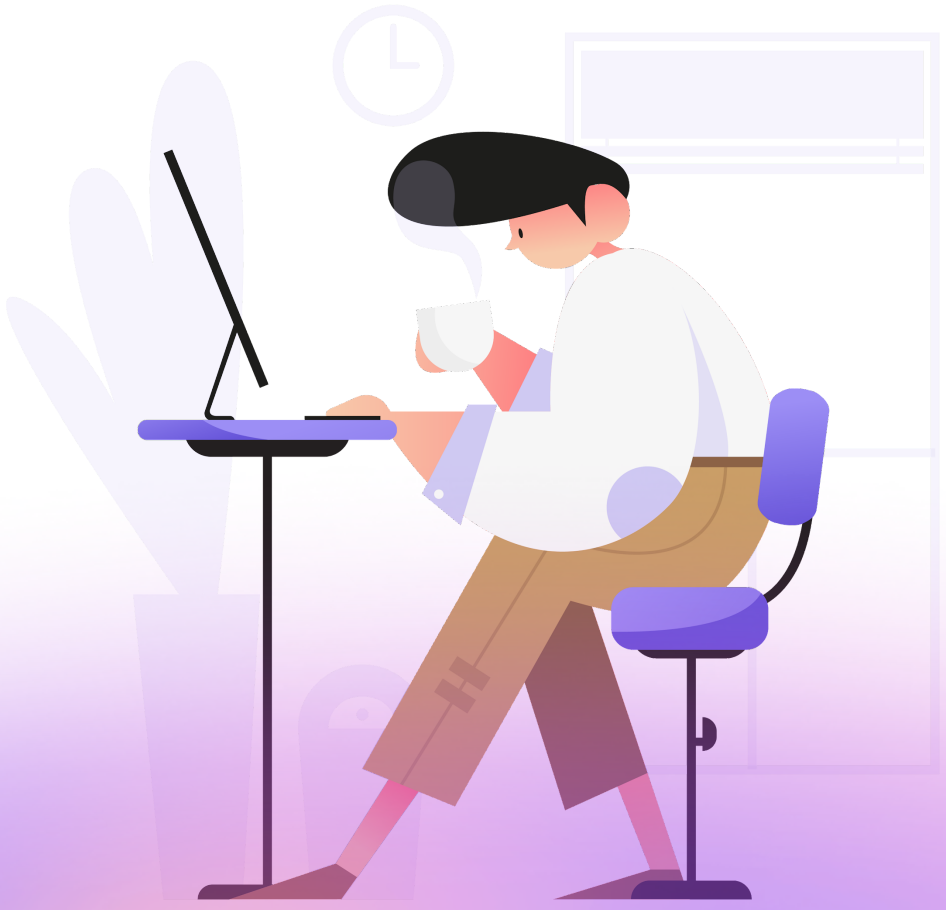
GeekBrains

Методы сбора и обработки данных при помощи Python

Открытые данные



На этом уроке

1. Узнаем, что такое открытые данные, для чего они нужны и как используются.
2. Научимся работать с CSV в Python и самостоятельно их создавать.

Оглавление

Open data

Концепция открытых данных основана на идее, что некоторые данные должны быть открытыми, машиночитаемыми и доступными для последующего использования без каких-либо ограничений. Сейчас основной поставщик таких данных — государство, это отражается в особом документе — методических рекомендациях по публикации открытых данных. В общем смысле открытые данные — это часть информации определённого рода, раскрываемой органами государственной власти и местного самоуправления. Двигатель внедрения открытых данных в России — проект «Открытые данные». Предполагается, что на этом фундаменте должна будет вырасти система открытого правительства. Открытые данные имеют хороший потенциал как индикатор отображения различных показателей на государственном уровне.

Примеры открытых данных:

- государственный реестр лекарственных средств;
- результаты голосований на уровне посёлка/города/страны;
- данные вызовов экстренных служб по месяцам;
- список католических храмов в России;
- список зарегистрированных Wi-Fi точек в городе.

На основе таких данных можно построить множество отчётов на разных уровнях, подсчитать «индекс Биг Мака» в разных странах, отразить «пульс» города или страны, выявить проблемы и понять, как лучше их решить.

Вот несколько известных источников открытых данных:

- https://proverki.gov.ru/opendata/;
- http://data.gov.ru;

- http://open.gov.ru;
- http://data.mos.ru;

Типичные проблемы открытых данных — невалидность и разрозненность. Причины этого — данных огромное количество, они очень разнородны, а методические рекомендации по их подготовке и выдаче соблюдаются редко.

Пример плохого CSV-файла открытых данных:

```
regnumber,regdate,enddate,cancellationdate,nameregcertificate,country,tradename,
internationalname,formrelease,stages,barcodes,normativedocumentation,pharmacothe
rapeuticgroup
,,,,"ООО ""Валеант""",Россия,,,,"Производство готовой лекарственной
формы,Херкель Б.В., Nobelweg 6, 3899 BN Zeewolde, the Netherlands, Нидерланды
",,"П N009886-280411,2011,Бронхинол;
",отхаркивающее средство растительного происхождения
,,21.10.2010,,Дарница фармацевтическая фирма ЗАО,Украина,,,"таблетки 200 мг,
упаковки ячейковые контурные - 2
","Производитель (Все стадии производства),Дарница фармацевтическая фирма ЗАО,
~, Украина
","4823006400096,Дарница фармацевтическая фирма ЗАО,пред-ль в Москве ОАО
""Бофарм"": 141191, Московская обл., г. Фрязино, ул. Станционная, д. 2, корп.
1,Украина
","НД 42-7644-04,2004,ПЕНТОКСИФИЛЛИН-ДАРНИЦА;
",вазодилатирующее средство
,,21.10.2010,,Дарница фармацевтическая фирма ЗАО,Украина,,,"концентрат для
приготовления раствора для внутривенного и внутриартериального введения 20
мг/мл, ампулы - 5
","Производитель (Все стадии производства),Дарница фармацевтическая фирма ЗАО,
~, Украина
","4823006400256,Дарница фармацевтическая фирма ЗАО,пред-ль в Москве ОАО
""Бофарм"": 141191, Московская обл., г. Фрязино, ул. Станционная, д. 2, корп.
1,Украина
","НД 42-7644-04,2004,ПЕНТОКСИФИЛЛИН-ДАРНИЦА;
",вазодилатирующее средство
,,16.09.2010,01.10.2010,Галичфарм АО,Украина,,,"раствор для внутривенного
введения 0.25 мг/мл, ампулы - 10
","Производитель (Все стадии производства),Галичфарм АО, г. Львов, ул.
Опрышковская, 6/8, Украина
","4823000800250,Галичфарм АО,г. Львов, ул. Опрышковская, 6/8,Украина
","НД 42-9049-05,2005,СТРОФАНТИН К;
",кардиотоническое средство - сердечный гликозид
,,,,"АО ""Галичфарм""",Украина,,,"раствор для внутривенного и внутримышечного
введения 0.2500 мг/мл, ампулы из прозрачного бесцветного стекла - 10
","Все стадии,АО ""Галичфарм"", , Украина
","4823000800250,АО ""Галичфарм"",,Украина
","010675-200911,2011,;
",
,,,12.03.2008,Московская фармацевтическая фабрика
ОАО,Россия,,,"субстанция-настойка гомеопатическая матричная ~, флаконы темного
стекла -
","Производитель (Все стадии производства),Московская фармацевтическая фабрика
ОАО, ~, Россия
```

```
",,"ВФС 42-3628-00,2000,НАСТОЙКА ЗВЕРОБОЯ ГОМЕОПАТИЧЕСКАЯ (""ГИПЕРИКУМ 1 Д"");
",гомеопатическое монокомпонентное средство
,,,,Опытный завод АН Республики Башкортостан,Россия,,,"субстанция-порошок ~,
пакеты полиэтиленовые двухслойные -
",,"~,Опытный завод АН Республики Башкортостан,~,Россия
",,
```

Принципы и методы построения

По статистике, более 70% открытых данных размещены в формате CSV (Comma-Separated Values — значения, разделённые запятыми), при том, что машиночитаемыми могут быть данные в следующих форматах:

- CSV;
- XML;
- JSON;
- HTML+RDFa;
- HTML+Microdata.

CSV — это текстовый формат, предназначенный в основном для табличных данных, так как в нём каждая строка текстового файла представляет собой строку в таблице. Разделитель по умолчанию — запятая, но на практике очень часто можно встретить и другие символы в качестве разделителей (точку с запятой, двоеточие, табуляцию). Пример файла формата CSV:

```
id, level,phrase1,phrase2,numcode
90,11,"hello","world",1
91,235,"si\"3\"sta","kingdom",300
92,0,"couch",,84
```

Первая строка — это обозначение колонок. Далее в каждой строчке следуют значения через разделитель (в данном случае это запятая). Все строки заключены в двойные кавычки, если такие же кавычки встречаются в тексте, их необходимо экранировать, как это сделано в записи с id=91 в колонке phrase1. Если значение отсутствует, то в строке будет пустое место (не пробел), за которым обязательно должен следовать разделитель (иначе нарушится структура), как это видно в строке id=92: в колонке phrase2 нет значения.

К открытым данным предъявляются следующие требования:

- портал открытых данных должен быть доступен по следующему адресу: http://domain.com/opendata; Error 404
- набор открытых данных включает в себя сами открытые данные (файл) и метаинформацию, состоящую из:
    - паспорта набора;
    - структуры набора (csv, xsd, json-schema);
    - иной информации, описывающей набор открытых данных, куда входят:
        - статическая информация;
        - лицензия;
        - человеческое отображение и машиночитаемое представление данных;
- реестр (т. е., список) наборов открытых данных должен иметь машиночитаемое представление и быть доступным по адресу http://domain.com/opendata/list.xml (в случае, если для формата данных выбран XML).

Паспорт открытых данных имеет чёткую структуру:

1. Идентификационный номер.
2. Наименование набора открытых данных.
3. Описание набора открытых данных.
4. Владелец набора открытых данных.
5. Ответственное лицо.
6. Телефон ответственного лица.
7. Адрес электронной почты ответственного лица.
8. Гиперссылка (URL) на открытые данные.
9. Формат набора открытых данных.
10. Описание структуры набора открытых данных.
11. Дата первой публикации набора открытых данных.
12. Дата последнего внесения изменений.
13. Содержание последнего изменения.
14. Дата актуальности набора данных.
15. Ключевые слова, соответствующие содержанию набора данных.
16. Гиперссылки (URL) на версии открытых данных.
17. Гиперссылки (URL) на версии структуры набора данных.
18. Версия методических рекомендаций.

Идентификационный номер формируется из следующих параметров:

- формат идентификационного номера: <код организации>-<наименование набора>;
- код организации представляет собой идентификационный номер налогоплательщика (ИНН), соответствующий государственному органу, органу местного самоуправления или организации, опубликовавшей набор открытых данных;
- наименование набора открытых данных — сокращённое англоязычное название набора открытых данных, указывается в одно слово (уникальное в пределах организации).

Пример: 7712345678-inspections.

Наборы открытых данных могут быть оперативными — обновляться чаще, чем раз в неделю, и долговременными — реже, чем раз в неделю.

Работа с open data в Python

Так как большая часть open data представлена в формате CSV, именно с ним мы будем работать в Python. Для работы с CSV в стандартной поставке Python есть библиотека csv.

Файл формата CSV, с которым будем работать:

```
"Grade","Ounce","Gram","Inch","mm","PPO"
"#TriBall",0.7199,20.41,0.60,15.24,1
"#0000",0.1943,5.51,0.38,9.40,5
"#000",0.1601,4.54,0.36,9.14,6
"#00",0.1231,3.49,0.33,8.38,8
"#0",0.1122,3.18,0.32,8.13,9
"#1",0.0924,2.62,0.30,7.62,11
"#2",0.0674,1.91,0.27,6.86,15
"#3",0.0536,1.52,0.25,6.35,19
"#4",0.0473,1.34,0.24,6.09,21
"#FF",0.0416,1.18,0.23,5.84,24
"#F",0.0370,1.05,0.22,5.59,27
"#TT",0.0346,0.98,0.21,5.33,29
"#T",0.0314,0.89,0.20,5.08,32
"#BBB",0.0233,0.66,0.19,4.82,43
"#BB",0.0201,0.57,0.18,4.57,50
"#B",0.0169,0.48,0.17,4.32,59
"2",0.0109,0.31,0.148,3.76,92
"4",0.0071,0.20,0.129,3.28,142
"5",0.0060,0.17,0.120,3.05,167
"6",0.0042,0.12,0.109,2.77,236
"7.5",0.0028,0.078,0.094,2.39,364
"8",0.0023,0.066,0.089,2.26,430
"8.5",0.0020,0.058,0.085,2.16,489
"9",0.0017,0.047,0.079,2.01,603
"12",0.0005,0.014,0.050,1.30,2025
```

Построчно читаем CSV файл:

```
import csv
csv_path = /path/to/lesson_7.csv'
with open(csv_path, newline='') as csvfile:
      reader = csv.DictReader(csvfile)
    field_names = reader.fieldnames
    print(f"Field Names: {field_names}")
      for row in reader:
      print(f"Row: {row}")
      print(f"Access to row element: {row['Ounce']}")
```

Вывод:

```
Field Names: ['Grade', 'Ounce', 'Gram', 'Inch', 'mm', 'PPO']
Row: OrderedDict([('Grade', '#TriBall'), ('Ounce', '0.7199'), ('Gram', '20.41'),
('Inch', '0.60'), ('mm', '15.24'), ('PPO', '1')])
Access to row element: 0.7199
Row: OrderedDict([('Grade', '#0000'), ('Ounce', '0.1943'), ('Gram', '5.51'),
('Inch', '0.38'), ('mm', '9.40'), ('PPO', '5')])
Access to row element: 0.1943
Row: OrderedDict([('Grade', '#000'), ('Ounce', '0.1601'), ('Gram', '4.54'),
('Inch', '0.36'), ('mm', '9.14'), ('PPO', '6')])
Access to row element: 0.1601
```

```
Row: OrderedDict([('Grade', '#00'), ('Ounce', '0.1231'), ('Gram', '3.49'),
('Inch', '0.33'), ('mm', '8.38'), ('PPO', '8')])
Access to row element: 0.1231
Row: OrderedDict([('Grade', '#0'), ('Ounce', '0.1122'), ('Gram', '3.18'),
('Inch', '0.32'), ('mm', '8.13'), ('PPO', '9')])
..
```

Добавим строку в существующий файл:

```
import csv
csv_path = /path/to/lesson_7.csv'
with open(csv_path, 'a', newline='') as csvfile:
    file_writer = csv.writer(csvfile, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
    writeable_row = ["12345test", 1.2345, 6.78, 9.012, 3.45, 67]
    file_writer.writerow(writeable_row)
Создадим новый CSV-файл и прочитаем его:
 new_csv_path = /path/to/lesson_7_new_file.csv'
with open(new_csv_path, 'w', newline='') as csvfile:
    fieldnames = ['id1', 'id2', 'id3', 'id4', 'id5', 'id6', 'id7', 'id8', 'id9',
'string']
    file_writer = csv.DictWriter(csvfile, fieldnames=fieldnames, delimiter=",",
quoting=csv.QUOTE_MINIMAL)
    file_writer.writeheader()
    for i in range(10):
        writeable_row = {
            'id1': 1,
            'id2': 2,
            'id3': 3,
            'id4': 4,
            'id5': 5,
            'id6': 6,
            'id7': 7,
            'id8': 8,
            'id9': 9,
            'string': f"ne\"w te\\:st string {i}"
            }
        file_writer.writerow(writeable_row)
with open(new_csv_path, newline='') as csvfile:
    reader = csv.DictReader(csvfile)
    field_names = reader.fieldnames
    print(f"Field Names: {field_names}")
    for row in reader:
        print(f"Row: {row}")
        print(f"Access to row element: {row['id5']}")
```

Вывод:

```
Field Names: ['id1', 'id2', 'id3', 'id4', 'id5', 'id6', 'id7', 'id8', 'id9',
'string']
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 0')])
Access to row element: 5
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 1')])
Access to row element: 5
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 2')])
Access to row element: 5
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 3')])
Access to row element: 5
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 4')])
Access to row element: 5
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 5')])
Access to row element: 5
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 6')])
Access to row element: 5
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 7')])
Access to row element: 5
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 8')])
Access to row element: 5
Row: OrderedDict([('id1', '1'), ('id2', '2'), ('id3', '3'), ('id4', '4'),
('id5', '5'), ('id6', '6'), ('id7', '7'), ('id8', '8'), ('id9', '9'), ('string',
'ne"w te\\:st string 9')])
Access to row element: 5
```

Получившийся файл:

```
id1,id2,id3,id4,id5,id6,id7,id8,id9,string
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 0"
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 1"
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 2"
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 3"
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 4"
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 5"
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 6"
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 7"
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 8"
1,2,3,4,5,6,7,8,9,"ne""w te\:st string 9"
```

Рассмотрим практический пример с использованием open data. Для этого возьмём CSV и двумя разными способами получим из него данные. Сначала подключим библиотеки pprint, csv и get. Для первого способа получить доступ к CSV за пределами проекта невозможно, поэтому мы его скачаем к себе в папку проекта (через get-запрос) и сохраним как data.csv, после чего уже будем с ним работать.

```
url =
'https://data.gov.ru/opendata/7706562710-harakteristika/data-20200113T1400-struc
ture-20200113T1400.csv'
data = get(url)
f = open('data.csv','wb')
f.write(data.content)
f.close()
```

Откроем наш файл для чтения, укажем кодировку и загрузим в класс DictReader. Обязательно нужно указать параметр delimiter — он указывает, какой знак у нас будет разделителем. После этого для вывода результата на экран создадим field_names, укажем команду print и перечислим, какие именно столбцы мы хотим получить на экране.

```
from pprint import pprint
import csv
from requests import get
import pandas as pd
# url =
'https://data.gov.ru/opendata/7706562710-harakteristika/data-20200113T1400-str
ucture-20200113T1400.csv'
# data = get(url)
# f = open('data.csv','wb')
# f.write(data.content)
# f.close()
  with open('data.csv','r',encoding='UTF-8') as f:
      reader = csv.DictReader(f, delimiter=',')
      field_names = reader.fieldnames
      print(field_names)
      for row in reader:
      print(row["годы"],row["количество осужденных за убийство"],row["количество
осужденных за кражу"])
```

Для второго способа подключим библиотеку pandas и создадим data_frame из нашего CSV-файла, вызывая соответствующий метод:

```
pd.read_csv(url,sep=',')
```

Отличительная особенность этого способа — возможность читать данные напрямую по ссылке. Вместо delimiter в pandas используется параметр sep — укажем разделителем запятую, как и в первом способе. После этого настроим вывод информации на экран.

```
from pprint import pprint
import csv
from requests import get
import pandas as pd
url =
'https://data.gov.ru/opendata/7706562710-harakteristika/data-20200113T1400-struc
ture-20200113T1400.csv'
data_frame = pd.read_csv(url,sep=',')
result = data_frame[data_frame['годы'] > 2015]
print(result)
```

Создание своего CSV-файла с помощью Scrapy

Мы уже убедились, что правильный, валидный CSV-файл — это залог успешной работы с набором открытых данных. Такие файлы вручную не создаются и мы можем воспользоваться уже имеющимися у нас знаниями, чтобы автоматизировать этот процесс. Вернёмся к нашему проекту Scrapy по сбору фотографий с Avito.ru. Откроем файл pipelines и подключим в нём модуль для работы с CSV-файлами:

```
import csv
```

Создадим в нём новый класс pipeline:

```
class CSVPipeline():
    def __init__(self):
        self.file = f'database.csv'
        with open(self.file, 'r', newline='') as csv_file:
            self.tmp_data = csv.DictReader(csv_file).fieldnames

        self.csv_file = open(self.file, 'a', newline='', encoding='UTF-8')

    def __del__(self):
        self.csv_file.close()

    def process_item(self, item, spider):
        columns = item.fields.keys()

        data = csv.DictWriter(self.csv_file, columns)
        if not self.tmp_data:
```

```
            data.writeheader()
            self.tmp_data = True
        data.writerow(item)
        return item
```

Внимание!

Мы должны заранее позаботиться о том, чтобы у нас был создан пустой CSV-файл, имя которого мы указываем в конструкторе класса pipeline!

Сначала мы определяем конструктор:

```
def __init__(self):
```

Внутри него задаём свойство self.name — имя созданного CSV-файла. После этого открываем файл на чтение, считываем его содержимое с помощью метода csv.DictReader() и сразу же заполняем свойство tmp_data результатом, полученным из свойства fieldnames объекта DictReader:

```
with open(self.file, 'r', newline='') as csv_file:
            self.tmp_data = csv.DictReader(csv_file).fieldnames
```

Затем снова открываем файл на чтение, но уже без оператора with. Это нужно сделать, потому что нам необходимо держать файл открытым до тех пор, пока существует наш класс:

```
self.csv_file = open(self.file, 'a', newline='', encoding='UTF-8')
```

Далее в методе process_item мы считываем значения всех ключей пришедшего item, чтобы определить заголовки столбцов для нашего файла CSV:

```
def process_item(self, item, spider):
        columns = item.fields.keys()
```

Создаём объект DictWriter, чтобы иметь возможность записывать данные в файл:

```
data = csv.DictWriter(self.csv_file, columns)
```

Дальше нам понадобится то самое свойство, куда мы считали fieldnames объекта DictReader в конструкторе, чтобы определить, пустой ли наш файл или в нём уже есть данные. Это требуется, чтобы понять, нужно ли нам создавать строку с наименованием столбцов или можно дописать новые данные в файл:

```
if not self.tmp_data:
            data.writeheader()
            self.tmp_data = True
```

После записываем пришедшие данные из item в файл:

```
data.writerow(item)
```

Не забываем, что наш файл необходимо закрыть, когда он перестанет существовать. Для этого определим деструктор, код которого будет выполняться в момент уничтожения класса и закроем там файл:

```
    def __del__(self):
        self.csv_file.close()
```

Глоссарий

Открытые данные — концепция, отражающая идею о том, что определённые данные должны быть свободно доступны для машиночитаемого использования и дальнейшей републикации без ограничений авторского права, патентов и других механизмов контроля.

CSV — текстовый формат, предназначенный для представления табличных данных. Строка таблицы соответствует строке текста, которая содержит одно или несколько полей, разделённых запятыми.

Паспорт открытых данных — совокупность сведений о наборе открытых данных, необходимых для установления факта их принадлежности к той или иной тематической рубрике, его потенциальной пригодности для решения задач потребителя, а также установления адреса размещения, способа загрузки и последующей автоматической обработки набора открытых данных..

Машиночитаемые данные — данные, представленные в описанном формате, позволяющем информационным системам без участия человека идентифицировать, обрабатывать, преобразовывать такие данные и их составные части (элементы), а также обеспечивать доступ к ним.

Дополнительные материалы

1. Обрабатываем csv файлы — модуль csv.

Домашнее задание

1. В ранее написанное приложение добавить класс с функциями, которые позволят собрать открытые данные по выбранной теме при помощи Python с сайта (выберите из списка известных источников данных).

Используемая литература

1. CSV File Reading and Writing.
2. Reading CSV files in Python.
3. Открытые данные.

4. Почему открытые данные никому не нужны.
5. Реестр наборов открытых данных.
6. Портал открытых данных.
7. Википедия. Открытые данные.
8. EU Open data.