

Лекция 9. Работа с файлами

Открытие, чтение и запись текстовых файлов.

Проблема кодировок.

Форматы json, csv, xml, pickle





Текстовые файлы

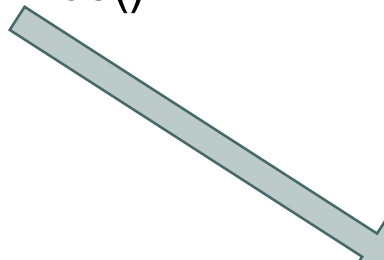
Открытие и чтение текстовых файлов.

Проблема кодировок

Чтение и запись текстовых файлов

Чтение:

- ☐ Построчное for line in file
- ☐ Полное - через file.readlines()
в список строк



```
lines_count = 0
with open("files/uav.csv") as f:
    for line in f:
        print(line)
        lines_count += 1
print(f"Всего в файле {lines_count} строк")
```

Запись:

- ☐ Построчная – через file.write()

```
with open("files/uav.txt", "w") as f:
    for line in lines:
        f.write(line)
```

```
with open("files/uav.csv") as f:
    lines = f.readlines()
    for line in lines:
        print(line)
print(f"Всего в файле {len(lines)} строк")
```



Проблема кодировок

Зачастую Python неправильно самостоятельно определяет кодировку. Особенно кириллицу. Для решения проблемы при открытии файла кодировка задается в параметре **encoding**. Их две основных:

- ❑ **cp1251** (windows 1251) – однобайтная, часто встречается как наследие выгрузки из Excel
- ❑ **utf-8** (Unicode utf-8) – многобайтная, обычно это кодировка по умолчанию

```
with open("files/uav.csv", encoding="utf-8") as f:  
    lines = f.read().split("\n")
```





Файлы CSV

Чтение и запись файлов CSV.

Настройка диалекта.

Избавление от пустых строк

Чтение и запись CSV

Чтение CSV:

- ❑ `csv.reader` – построчно в list
- ❑ `csv.DictReader` – построчно в dict
- ❑ `list(csv.reader)` – файл целиком в список списков [list, list, ...]

Запись CSV:

- ❑ `csv.writer.writerow()` – по одной строке (по одному list)
- ❑ `csv.writer.writerows()` – все содержимое сразу ([list, list, ...])



Три способа чтения CSV (1)

Построчно через reader

Одна запись = list

Плюсы:

- ☐ можно читать файлы любого размера
- ☐ малый расход памяти

Минусы:

- ☐ читает по одной записи за проход
- ☐ нельзя получить статистику сразу
- ☐ чтобы обратиться к элементу в list, нужно знать его порядковый номер

```
with open("files/uav.csv") as f:  
    reader = csv.reader(f)  
    lines_list = list(reader)  
print(lines_list[:3])
```



Три способа чтения CSV (2)

Целиком через преобразование в list

Весь файл = [list, list, ...]

Плюсы:

- ☐ Читает файл целиком
- ☐ Можно получить всю статистику сразу

Минусы:

- ☐ Не подходит для больших файлов
- ☐ Большой расход памяти (весь файл помещается в память)
- ☐ Чтобы обратиться к элементу в list, нужно знать его порядковый номер

```
import csv

with open("files/uav.csv") as f:
    reader = csv.reader(f)
    for line in reader:
        print(line)
```



Три способа чтения CSV (3)

Построчно через DictReader

Одна запись = dict

Плюсы:

- ☐ Можно читать файлы любого размера
- ☐ Малый расход памяти
- ☐ К элементу можно обратиться по имени вместо индекса

Минусы:

- ☐ читает по одной записи за проход
- ☐ нельзя получить статистику сразу

```
with open("files/uav.csv") as f:
    reader = csv.DictReader(f)
    for line in reader:
        print(line["Timestamp"],
              line["Drone Make & Model:])
```



Запись CSV

Открыть файл:

на запись "w", на дозапись "a"

Построчно через writerow

Одна запись = list

Все содержимое сразу через writerows

Одна запись = [list, list, ...]

```
with open("files/uav2.csv", "w") as f:  
    writer = csv.writer(f)  
    for line in lines_list:  
        writer.writerow(line)
```

```
with open("files/uav2.csv", "w") as f:  
    writer = csv.writer(f)  
    writer.writerows(lines_list)
```



Настройки CSV и диалект

```
delimiter=","  
quoting=csv.QUOTE_MINIMAL (QUOTE_ALL, QUOTE_NONNUMERIC, QUOTE_NONE)  
quotechar=""  
escapechar="\"
```

```
csv.register_dialect("semicol_no_escape", delimiter=";",  
quoting = csv.QUOTE_NONE, escapechar="\\")
```

```
with open("files/uav2.csv", "w") as f:  
    writer = csv.writer(f, "semicol_no_escape")  
    writer.writerows(lines_list)
```



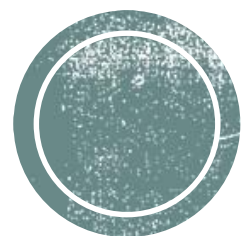
Пустые строки

Для того чтобы в CSV не записывались пустые строки (например, если в конце строк было записано несколько символов переноса строки), можно это запретить через параметр **newline=""**

Внимание: для работы с текстовыми файлами параметр `newline` включать нельзя, т.к. при этом пропадет возможность построчного чтения файла

```
with open("files/uav.csv", newline="") as f:
    reader = csv.reader(f)
    for line in reader:
        print(line)
```





Файлы JSON

Чтение и запись JSON.

Кириллица в JSON и форматирование

Чтение и запись JSON

Чтение JSON:

- ❑ `load()` – из файла
- ❑ `loads()` – из строки

Запись JSON:

- ❑ `dump()` – в файл
- ❑ `dumps()` – в строку



Чтение и запись JSON (продолжение)

- ❑ Для более удобного вывода JSON используйте pprint
- ❑ При сохранении JSON ставьте флаг ensure_ascii=True, чтобы не-латиница осталась буквами
- ❑ Для сохранения в читаемом варианте (с отступами) ставьте флаг indent=размер_отступа

```
with open("files/open_meteo_api.json") as f:
    json_data = json.load(f)
pprint(json_data)

with open("files/open_meteo_api2.json", "w") as f:
    json.dump(json_data, f, ensure_ascii=False, indent=2)
```





Файлы XML

Чтение XML, навигация по дереву

Кириллица в XML

Запись XML – обычная и с форматированием

Чтение XML

- ❑ XML – это дерево. Можно считать его вложенными списками
- ❑ К элементам внутри XML-дерева можно обращаться по индексам
- ❑ Чтобы работать с XML, нужно получить «корень» - root

```
import xml.etree.ElementTree as ET
# если будет проблема с кодировкой, сначала нужно создать парсер и явно
# задать кодировку файла
parser = ET.XMLParser(encoding="utf-8")
# подставляем парсер при разборе xml
tree = ET.parse("files/drone_news.xml", parser)
# что такое корневой элемент xml
root = tree.getroot()
```



Чтение XML (продолжение)

```
<Тег><channel> Текст
<Тег> <title>DRONELIFE</title> Атрибут Атрибут Атрибут
<Тег> <ns0:link href="https://dronelife.com/feed/" rel="self" type="application/rss+xml">
    </ns0:link> Текст
<Тег> <link>https://dronelife.com/</link>
```

```
# тег (название элемента)
print(root.tag)
# атрибут ("настройки" элемента)
print(root.attrib)
# текст – то, что находится между открывающим и закрывающим тегом
print(root.text)
```



Чтение XML (продолжение)

- ❑ Для работы с элементами дерева используется язык запросов **Xpath**
- ❑ Основные методы: **find()**, **findall()**

```
news_list = root.findall("channel/item")
print(len(news_list))
for news in news_list:
    print(f'Заголовок: {news.find("title").text}')
    print(f'Содержание: {news.find("description").text}')

titles_list = root.findall("./*/*/title")
titles_list = root.findall("channel/item/title")
print(len(titles_list))
for title in titles_list:
    print(f'Заголовок: {title.text}')
```



Запись XML

Простая запись – `tree.write(название_файла)`

```
tree.write("files/drone_news2.xml")
```

«Красивая» запись требует библиотеку `xmlformatter` и немного кода. К сожалению, все типовые способы форматирования зависят от версии Python (форматирование поддерживается с версии 3.9, а в других случаях появляются пустые строки)

```
import xmlformatter
formatter = xmlformatter.Formatter(indent="2", indent_char=" ")
prettyxml = formatter.format_string(ET.tostring(root)).decode("utf-8")

with open("files/drone_news3.xml", "w") as f:
    f.write(prettyxml)
```





Файлы Pickle

Бинарные файлы.

Чтение и запись Pickle

Чтение и запись Pickle

Pickle – это двоичный объект, поэтому в него можно записать почти все, что угодно (в отличие от JSON)

```
with open("files/open_meteo_sample.pickle", "wb") as f:
    pickle.dump(test_data, f)

with open("files/open_meteo_sample.pickle", "rb") as f:
    json_data = pickle.load(f)
print(json_data)
```

