МЕТОДИЧКА

Урок 4 (2 пары по 80 минут): Работа с файлами, CSV и JSON

Этот урок продолжает тему структур данных (list/dict/set/tuple) и показывает, как сохранять данные на диск и загружать их обратно. Мы разберем текстовые файлы (.txt), табличный формат CSV и структурный формат JSON.

Главная идея: пока программа работает, данные лежат в памяти (list/dict). Как только программа закрылась - память очищается. Файл - это способ сохранить результат навсегда.

Что подготовить:

- Python 3.x установлен и запускается из VS Code или терминала.
- Папка проекта с подпапкой data/.
- Текстовый редактор (VS Code).

Структура проекта (рекомендуется):

```
project/
  main.py
  data/
    shop.txt
    visits.txt
    students.csv
    profile.json
```

1. Цели урока и результаты обучения

После урока студент умеет:

- открывать файлы разными режимами: r, w, a, x;
- читать данные из файла: read(), readline(), readlines(), for line in file;
- записывать данные в файл и понимать роль символа переноса строки \n;
- объяснять, что такое кодировка UTF-8 и почему появляются 'кракозябры';
- работать с CSV (таблица) и JSON (структура) стандартными модулями Python;
- превращать файл -> структуры данных и структуры данных -> файл.

Важно: модули csv и json - встроенные (стандартная библиотека). Их не нужно устанавливать через pip.

2. План урока по времени

Пара	Блок	Время	Содержание
1	Файлы TXT	80 мин	пути, open(), with, чтение/запись, UTF-8, практика (частоты
2	CSV и JSON	80 мин	csv.DictReader/Writer, типы данных, json.load/dump, отчет С

3. Пара 1: Текстовые файлы (.txt)

3.1 Путь к файлу и рабочая папка (cwd)

Когда вы пишете open('data/shop.txt'), Python ищет файл относительно рабочей папки (папка, из которой вы запускаете программу). Если файл существует, но лежит в другом месте, появится ошибка FileNotFoundError.

Типовая ошибка: файл 'shop.txt' есть на компьютере, но не в папке проекта.
Решение: положить файл в data/ или указать правильный путь.

3.2 Функция open() и режимы

Режим	Что делает	Когда использовать
r	Читает файл	когда файл уже существует, и надо получить данные
w	Перезаписывает файл (стирает старое)	когда надо сохранить новый результат
a	Добавляет в конец файла	логи, история действий, накопление данных
x	Создает новый файл, если его нет	когда важно не затереть существующий файл

3.3 Почему with обязателен

with гарантирует, что файл будет закрыт автоматически даже при ошибке. Это защищает от потери данных и проблем с доступом к файлу.

```
# правильный шаблон
with open("data/input.txt", "r", encoding="utf-8") as f:
    text = f.read()

print(text)
```

3.4 Чтение файла: 4 способа

Способ 1: read() - прочитать все сразу.

```
with open("data/input.txt", "r", encoding="utf-8") as f:
    text = f.read()
```

Способ 2: readline() - прочитать одну строку.

```
with open("data/input.txt", "r", encoding="utf-8") as f:
    line1 = f.readline()
    line2 = f.readline()
```

Способ 3: readlines() - список строк (list[str]).

```
with open("data/input.txt", "r", encoding="utf-8") as f:
    lines = f.readlines()  # список строк
```

Способ 4: цикл for line in file - лучший для больших файлов.

```
with open("data/input.txt", "r", encoding="utf-8") as f:
    for line in f:
        print(line.strip())  # strip() убирает \n и пробелы по краям
```

3.5 Запись в файл: write() и перенос строки \n

```
# w - перезаписать файл
```

```
with open("data/output.txt", "w", encoding="utf-8") as f:
    f.write("Line 1\n")
    f.write("Line 2\n")

# a - дописать в конец файла
with open("data/log.txt", "a", encoding="utf-8") as f:
    f.write("New log line\n")
```

3.6 Практика: частоты товаров (мини-проект)

Файл data/shop.txt содержит названия товаров, по одному в строке. Нужно: посчитать, сколько раз встречается каждый товар, найти самый популярный и сохранить отчет в data/report.txt.
Пример содержимого shop.txt:

```
milk
bread
milk
cola
bread
```

Решение (разбор построчно):

```
counts = {}  # словарь: товар -> сколько раз встретился

with open("data/shop.txt", "r", encoding="utf-8") as f:
    for line in f:
        item = line.strip()   # убираем \n и пробелы по краям
        if item == "":
            continue          # пропускаем пустые строки

        if item not in counts:
            counts[item] = 0  # если товар встретился впервые, создаем счетчик

        counts[item] += 1     # увеличиваем счетчик

# найти самый популярный товар вручную (пока без max)
best_item = None
best_count = -1

for item, c in counts.items():
    if c > best_count:
        best_count = c
        best_item = item

print("Частоты:", counts)
print("Топ товар:", best_item, best_count)

# сохранить отчет в файл
with open("data/report.txt", "w", encoding="utf-8") as f:
    f.write("=== REPORT ===\n")
    for item, c in counts.items():
        f.write(f"{item}: {c}\n")
    f.write(f"\nTOP: {best_item} ({best_count})\n")
```

3.7 Частые ошибки и как исправить

- FileNotFoundError: неправильный путь или нет папки data/.
- Пустые строки в выводе: забыли .strip() при чтении строк.
- Файл перезаписался: использовали режим 'w' вместо 'a'.

- Кракозябры: забыли encoding='utf-8' или файл в другой кодировке.
- Лишний перенос: line уже содержит \n, а print добавляет еще один.

4. Пара 2: CSV и JSON

4.1 Зачем нужны CSV и JSON

CSV - формат для таблиц (строки и колонки). Его удобно открывать в Excel и Google Sheets.
JSON - формат для хранения структур данных (словари и списки). Его используют сайты, API и приложения.

4.2 CSV: чтение таблицы как list[dict]

Пусть файл data/students.csv выглядит так:

```
name,age,grade
Ali,18,90
Diana,19,87
```

При чтении через csv.DictReader каждая строка превращается в словарь: {'name': 'Ali', 'age': '18', 'grade': '90'}. Обратите внимание: значения приходят строками.

```
import csv

students = []

with open("data/students.csv", "r", encoding="utf-8", newline="") as f:
    reader = csv.DictReader(f)
    for row in reader:
        # row: dict[str, str]
        row["age"] = int(row["age"])
        row["grade"] = int(row["grade"])
        students.append(row)

print(students)
```

4.3 CSV: запись таблицы из list[dict]

```
import csv

students = [
    {"name": "Ali", "age": 18, "grade": 90},
    {"name": "Diana", "age": 19, "grade": 87},
]

with open("data/out_students.csv", "w", encoding="utf-8", newline="") as f:
    fieldnames = ["name", "age", "grade"]
    writer = csv.DictWriter(f, fieldnames=fieldnames)
    writer.writeheader()
    writer.writerows(students)
```

Почему newline='' важно? На Windows без этого параметра иногда появляются пустые строки между записями.

4.4 JSON: чтение и запись структур данных

Пример JSON файла data/profile.json:

```
{
  "name": "Ali",
  "age": 18,
```

```
  "skills": ["python", "sql"]
}
```

Чтение JSON:

```
import json

with open("data/profile.json", "r", encoding="utf-8") as f:
    data = json.load(f)   # dict или list

print(data)
```

Запись JSON (красиво, с кириллицей):

```
import json

report = {
    "count": 2,
    "avg_grade": 88.5,
    "students": [
        {"name": "Ali", "age": 18, "grade": 90},
        {"name": "Diana", "age": 19, "grade": 87}
    ]
}

with open("data/students_report.json", "w", encoding="utf-8") as f:
    json.dump(report, f, ensure_ascii=False, indent=2)
```

4.5 Практика: CSV -> расчет -> JSON отчет

Нужно прочитать students.csv, посчитать количество студентов и среднюю оценку, затем сохранить структурированный отчет в JSON.

```
import csv, json

students = []

with open("data/students.csv", "r", encoding="utf-8", newline="") as f:
    reader = csv.DictReader(f)
    for row in reader:
        row["age"] = int(row["age"])
        row["grade"] = int(row["grade"])
        students.append(row)

count = len(students)
total = sum(s["grade"] for s in students)
avg_grade = total / count if count > 0 else 0

report = {
    "count": count,
    "avg_grade": round(avg_grade, 1),
    "students": students
}

with open("data/students_report.json", "w", encoding="utf-8") as f:
    json.dump(report, f, ensure_ascii=False, indent=2)
```

4.6 Дополнение: проценты (пример формата google: 3 (37.5%))

Если нужно вывести долю (процент) каждого элемента, используем формулу: percent = count / total * 100.

```
site = "google"
count = 3
total = 8

percent = round(count / total * 100, 1)
print(f"{site}: {count} ({percent}%)")  # google: 3 (37.5%)
```

4.7 Частые ошибки в CSV/JSON

- Забыли import csv/json: будет NameError.
- CSV все строками: если не сделать int(), средняя оценка сломается.
- Кириллица в JSON стала \u0410...: забыли ensure_ascii=False.
- Пустые строки в CSV на Windows: забыли newline=''.
- Неправильные названия колонок: fieldnames должны совпадать с заголовками CSV.

5. Домашнее задание

ДЗ 1 (база). ToDo список в файле

Файл data/todo.txt: каждая строка - одна задача. Написать программу, которая: 1) читает todo.txt в список (list) 2) спрашивает у пользователя новую задачу (input) 3) добавляет задачу в список 4) сохраняет обновленный список обратно в todo.txt (режим w) 5) выводит на экран сколько задач всего

Подсказка: при записи списка в файл используйте цикл и добавляйте \n после каждой задачи.

ДЗ 2 (средний). Анализ посещений с процентами

Файл data/visits.txt: каждая строка - название сайта. Нужно: 1) посчитать частоты (dict) 2) посчитать total = сумма всех посещений 3) вывести и сохранить отчет в формате: site: count (percent%) 4) найти top site и добавить строку TOP SITE: ...
Требование к формату одной строки:

```
google: 3 (37.5%)
```

ДЗ 3 (сложный). CSV фильтрация + новый CSV

Файл data/students.csv содержит столбцы name, age, grade. Нужно: 1) прочитать CSV 2) выбрать студентов с grade >= 85 3) записать их в data/good_students.csv с теми же заголовками 4) дополнительно вывести: сколько всего было и сколько осталось после фильтра

Критерии оценки

- программа запускается без ошибок и читает файлы из папки data/;
- правильно использован with open(...);
- есть обработка пустых строк (strip + continue);
- отчет сохранен в файл в требуемом формате;
- для CSV корректно приведены типы (int/float).

Мини-шпаргалка

```
# TXT
with open("data/file.txt", "r", encoding="utf-8") as f:
    for line in f:
        line = line.strip()

# CSV
import csv
with open("data/file.csv", "r", encoding="utf-8", newline="") as f:
    reader = csv.DictReader(f)

# JSON
import json
with open("data/file.json", "w", encoding="utf-8") as f:
    json.dump(data, f, ensure_ascii=False, indent=2)
```