ГЛЕБ ЗАЙЦЕВ

CSV И JSON В PYTHON

18 задач без Pandas

```
import csv, json, io

rows = csv.DictReader(...)

payload = json.loads(text)

result = clean(rows)
```

таблицы · вложенные поля · дедупликация

конвертация · проверки · отчёты

все примеры запускаются стандартной библиотекой

Глеб Зайцев

CSV и JSON в Python. 18 задач без Pandas

«Автор»
2026

Зайцев Г.

CSV и JSON в Python. 18 задач без Pandas / Г. Зайцев — «Автор», 2026

Практический мини-курс по обработке CSV, JSON и JSON Lines в Python без Pandas и внешних библиотек. Внутри 18 коротких задач: чтение таблиц, DictReader, очистка чисел, даты, группировки, сортировка, вложенный JSON, конвертация CSVJSON, дедупликация, объединение таблиц, валидация строк и потоковая обработка. Все данные встроены прямо в книгу, каждый пример можно скопировать и запустить локально.

Содержание

Глеб Зайцев
CSV и JSON в Python. 18 задач без Pandas

Перед первой выгрузкой

Кому пригодится эта книга

Эта книга для тех, кто уже написал первые скрипты на Python и столкнулся с обычными рабочими файлами: таблицей заказов, выгрузкой клиентов, JSON-ответом сервиса или логом в формате JSON Lines.

Мы не используем Pandas, базы данных и сетевые запросы. Это сознательное ограничение: сначала полезно понять, как устроены строки, словари, списки, даты и стандартные модули csv и json.

В каждом примере есть маленькие входные данные, цель, рабочий код, ожидаемый результат и короткое объяснение. Формат ближе к лабораторной тетради, чем к справочнику.

Как читать и проверять

Сначала прочитайте задачу и попробуйте предсказать форму результата: список словарей, число, строка отчёта или список ошибок. После этого запускайте код и сравнивайте вывод.

Если код кажется слишком простым, измените входные данные: добавьте пустую строку, лишний пробел, новый статус или отсутствующее поле. Именно такие мелочи чаще всего ломают реальные выгрузки.

Все фрагменты используют только стандартную библиотеку Python. Данные лежат в строках, поэтому для тренировки не нужно скачивать файлы и создавать аккаунты.

Маршрут по задачам

Восемнадцать задач

Задача 1. Посчитать сумму из CSV-строк

Рабочий вопрос

Есть выгрузка заказов: номер, клиент и сумма. Нужно получить общую выручку, не создавая файл на диске.

Код

```
import csv, io

data = '''order_id,client,total
1001,Анна,1290
1002,Илья,870
1003,Анна,410
'''

reader = csv.DictReader(io.StringIO(data))
result = sum(int(row['total']) for row in reader)
```

Ожидаемый результат

```
2570
```

Почему работает

io.StringIO превращает строку в объект, похожий на файл.
DictReader берёт заголовки первой строки и возвращает каждую запись словарём.
Суммировать удобнее после явного int: в CSV все значения приходят строками.

Где легко ошибиться

Частая ошибка — забыть преобразование типов и получить склейку строк или TypeError вместо арифметики.

Самостоятельная проверка

Добавьте четвёртый заказ на 330 рублей и пересчитайте итог.
Ответ: Если добавить строку 1004,Олег,330, результат станет 2900.

Задача 2. Оставить только оплаченные заказы

Рабочий вопрос

В таблице есть статусы. Нужно выбрать номера заказов со статусом paid и сохранить порядок строк.

Код

```
import csv, io

data = '''order_id,status,total
2001,paid,700
2002,cancelled,500
2003,paid,1250
2004,new,900
'''

reader = csv.DictReader(io.StringIO(data))
result = [row['order_id'] for row in reader if row['status'] == 'paid']
```

Ожидаемый результат

```
['2001', '2003']
```

Почему работает

Фильтр в списковом включении читается почти как условие задачи.
Номера заказов оставлены строками: с ними не выполняют арифметику.
Порядок сохраняется, потому что DictReader идёт сверху вниз.

Где легко ошибиться

Не смешивайте статус оплаты и сумму заказа: оплаченный заказ на маленькую сумму всё равно должен пройти фильтр.

Самостоятельная проверка

Измените статус заказа 2004 на paid. Какие номера должны остаться?
Ответ: Должны остаться ['2001', '2003', '2004'].

Задача 3. Подставить значение по умолчанию

Рабочий вопрос

В одной строке не заполнен город. Нужно заменить пустоту на 'не указан', чтобы отчёт не развалился.

Код

```
import csv, io

data = '''client,city
Анна,Москва
Илья,
Олег,Казань
'''

reader = csv.DictReader(io.StringIO(data))
result = []
for row in reader:
city = row['city'].strip() or 'не указан'
result.append((row['client'], city))
```

Ожидаемый результат

```
[('Анна', 'Москва'), ('Илья', 'не указан'), ('Олег', 'Казань')]
```

Почему работает

strip убирает пробелы вокруг значения.
Пустая строка в условии считается ложной.
Замена пустого значения делается рядом с чтением, чтобы дальше работать с чистыми данными.

Конец ознакомительного фрагмента.