Лабораторна робота 7

Тема: Текстові файли, CSV, JSON та обробка винятків.

Мета: Навчитися читати й записувати файли, опрацьовувати табличні записи та перевіряти результати після серіалізації.

Зміст роботи

Завдання 1. Короткий текстовий підсумок

Введіть шляхи вхідного й вихідного текстових файлів. Використайте UTF-8 і with. Файли мають бути різними.

1. Прочитайте текст, підрахуйте фізичні рядки та слова, розділені пробільними символами. Порожні рядки врахуйте в кількості рядків.
2. Виведіть показники й запишіть їх двома підписаними рядками у вихідний файл.
3. Окремо обробіть OSError під час читання й запису, повідомивши про невдалу операцію. Після невдалого читання не записуйте підсумок.

Перевірте власний текст, порожній файл і відсутній вхідний файл.

Таблиця 1 — Контрольна перевірка текстового файла

Вхідний файл або дія	**Очікуваний результат**	**Що перевіряється**
Три рядки: «Мова Python», порожній, «Код».	Рядків 3, слів 3. Два підписані рядки у вихідному файлі.	Порожній фізичний рядок
Порожній файл.	Рядків 0, слів 0.	Відсутність даних
Є лише «Код» без перенесення наприкінці.	Рядків 1, слів 1.	Останній рядок
Вхідного файла немає.	Повідомлення про невдале читання; вихідний файл не записується.	OSError читання
Недоступний шлях запису.	Повідомлення про невдалий запис.	OSError запису

Завдання 2. Індивідуальне опрацювання CSV

Прочитайте через csv.DictReader CSV у UTF-8 з роздільником «;» і заголовком name;group;a;b. Поле name — назва запису, group — назва його групи; обидва поля непорожні. Поля a і b — скінченні дійсні числа ≥ 0; їхній зміст, додаткові межі та обчислення визначає ваш варіант у таблиці 3.

1. Для кожного коректного запису перевірте умову відбору й обчисліть показник. Збережіть відібрані записи як словники з полями name, group, value, у початковому порядку.

2. Об’єднайте відібрані записи з однаковим значенням group в одну групу. Для кожної групи обчисліть підсумок value за таблицею 3: суму або арифметичне середнє. Виведіть показники записів, підсумки груп за алфавітом, кількість відібраних і некоректних записів.
3. Обробіть OSError читання, csv.Error та ValueError числового перетворення. Некоректний запис пропустіть із коротким повідомленням. За неправильного заголовка, OSError або csv.Error припиніть обробку й не переходьте до збереження JSON.

Крайні пробіли полів приберіть; регістр назв збережіть. Повністю порожні рядки ігноруйте. Рядок із зайвим чи відсутнім полем, порожньою назвою або недопустимим числом — некоректний. Коректний запис, що не пройшов відбір, не є помилкою. Відсутність відібраних дає [] і {}; числа друкуйте з двома десятковими знаками, у пам’яті не округлюйте. Шлях вводить користувач.

Для основної перевірки створіть CSV із наведеним нижче вмістом. Записи «Запис 1» і «Запис 3» мають спільну групу «Група 1»; «Запис 2» має групу «Група 2». Очікувані показники записів, підсумки груп і загальний підсумок подано в таблиці 3. У цьому файлі некоректних записів немає: skipped = 0. Перевірте також власні дані.

Вхідний файл для контрольної перевірки

```
name;group;a;b
Запис 1;Група 1;2;3
Запис 2;Група 2;4;2
Запис 3;Група 1;1;5
```

Лише для варіанта 12 замініть значення a у цих трьох рядках на 2000, 4000 і 1000 Вт відповідно. Заголовок та інші поля залиште без змін.

Для строгого читання застосуйте strict=True. Помилка синтаксису CSV зупиняє обробку; некоректне числове поле спричиняє пропуск одного запису. Середнє — сума value, поділена на кількість відібраних записів групи або всього набору. Для порожнього набору приймаємо 0.

Кожну перевірку таблиці 2 виконуйте окремо. Для випадків із додаванням рядка щоразу беріть початковий контрольний файл.

Таблиця 2 — Спільна контрольна перевірка CSV

Вхідні дані або дія	**Очікуваний результат**	**Що перевіряється**
Лише заголовок name;group;a;b.	Відібрано записів: 0. Некоректних записів: 0. Підсумків груп немає.	Порожній набір даних
Додати рядок: Запис 4;Група 1;помилка;2 У полі a навмисно введено текст.	Повідомлення про некоректне число в полі a. Запис пропущено; некоректних записів: 1. Читання продовжується.	Некоректне числове поле
Додати один із рядків: Запис 4;Група 1;2 або Запис 4;Група 1;2;3;зайве	Повідомлення про неправильну кількість полів. Запис пропущено; некоректних записів: 1. Читання продовжується.	Три або п’ять полів замість чотирьох
Додати порожній фізичний рядок.	Рядок ігнорується. Кількість некоректних записів не змінюється.	Порожній рядок
Неправильний заголовок або помилка читання CSV.	Повідомлення про помилку; обробку зупинено. До збереження JSON програма не переходить.	Помилка всього файла

Таблиця 3 — Спільні варіанти опрацювання CSV та JSON

Варіант	**Поля та межі**	**Відбір**	**Показник і підсумок**	**Очікувані результати**
1	a — ціна, грн; b — ціла кількість.	$b \geq 3$	a × b, грн. Для груп і всіх записів: сума.	Запис 1: 6.00 Запис 3: 5.00 Група 1 — сума: 11.00 Загальна сума: 11.00 Відібрано записів: 2
2	a — обсяг; b — ставка, грн за одиницю.	$a \geq 2$	a × b + b, грн. Для груп і всіх записів: сума.	Запис 1: 9.00 Запис 2: 10.00 Група 1 — сума: 9.00 Група 2 — сума: 10.00 Загальна сума: 19.00 Відібрано записів: 2
3	a — дохід, b — витрати, грн.	$a > b$	a − b, грн. Для груп і всіх записів: сума.	Запис 2: 2.00 Група 2 — сума: 2.00 Загальна сума: 2.00 Відібрано записів: 1
4	a — план > 0; b — виконання.	$b \geq a$	100 × b / a, %. Для груп і всіх записів: середнє.	Запис 1: 150.00 Запис 3: 500.00 Група 1 — середнє: 325.00 Загальне середнє: 325.00 Відібрано записів: 2

Варіант	Поля та межі	Відбір	Показник і підсумок	Очікувані результати
5	a — відстань, км; b — час, год > 0.	a ≥ 2	a / b, км/год. Для груп і всіх записів: середнє.	Запис 1: 0.67 Запис 2: 2.00 Група 1 — середнє: 0.67 Група 2 — середнє: 2.00 Загальне середнє: 1.33 Відібрано записів: 2
6	a — ціна, грн; b — відсоток ≤ 100.	b ≥ 3	a × (1 − b / 100), грн. Для груп і всіх записів: сума.	Запис 1: 1.94 Запис 3: 0.95 Група 1 — сума: 2.89 Загальна сума: 2.89 Відібрано записів: 2
7	a, b — оцінки ≤ 100.	a + b ≥ 5	(a + b) / 2. Для груп і всіх записів: середнє.	Запис 1: 2.50 Запис 2: 3.00 Запис 3: 3.00 Група 1 — середнє: 2.75 Група 2 — середнє: 3.00 Загальне середнє: 2.83 Відібрано записів: 3
8	a, b — сторони, м > 0.	b > a	a × b, м². Для груп і всіх записів: сума.	Запис 1: 6.00 Запис 3: 5.00 Група 1 — сума: 11.00 Загальна сума: 11.00 Відібрано записів: 2
9	a, b — маси, кг; a + b > 0.	b ≥ a	100 × a / (a + b), %. Для груп і всіх записів: середнє.	Запис 1: 40.00 Запис 3: 16.67 Група 1 — середнє: 28.33 Загальне середнє: 28.33 Відібрано записів: 2
10	a — год; b — ставка, грн/год.	a ≥ 2	a × b; при a ≥ 4 додати 0.5 × b, грн. Для груп і всіх записів: сума.	Запис 1: 6.00 Запис 2: 9.00 Група 1 — сума: 6.00 Група 2 — сума: 9.00 Загальна сума: 15.00 Відібрано записів: 2
11	a, b — показники в одній одиниці.	|a − b| ≥ 2	|a − b|. Для груп і всіх записів: середнє.	Запис 2: 2.00 Запис 3: 4.00 Група 1 — середнє: 4.00 Група 2 — середнє: 2.00 Загальне середнє: 3.00 Відібрано записів: 2
12	a — потужність, Вт; b — час, год.	b ≥ 2	a × b / 1000, кВт·год. Для груп і всіх записів: сума.	Запис 1: 6.00 Запис 2: 8.00 Запис 3: 5.00 Група 1 — сума: 11.00 Група 2 — сума: 8.00 Загальна сума: 19.00 Відібрано записів: 3

Варіант	Поля та межі	Відбір	Показник і підсумок	Очікувані результати
13	a, b — сторони, м > 0.	a ≤ b	2 × (a + b), м. Для груп і всіх записів: сума.	Запис 1: 10.00 Запис 3: 12.00 Група 1 — сума: 22.00 Загальна сума: 22.00 Відібрано записів: 2
14	a — вміст, b — тара, кг; a + b > 0.	b ≥ 3	a + b, кг. Для груп і всіх записів: сума.	Запис 1: 5.00 Запис 3: 6.00 Група 1 — сума: 11.00 Загальна сума: 11.00 Відібрано записів: 2
15	a, b — оцінки ≤ 100.	b > a	0.4 × a + 0.6 × b. Для груп і всіх записів: середнє.	Запис 1: 2.60 Запис 3: 3.40 Група 1 — середнє: 3.00 Загальне середнє: 3.00 Відібрано записів: 2

Завдання 3. Збереження й перевірка JSON

Продовжіть варіант завдання 2 за таблицею 3. Збережіть словник із ключами records, totals, skipped: відібрані записи, підсумки груп і кількість некоректних записів. Назва totals є спільною для всіх варіантів, зокрема для середніх значень.

1. Введіть шлях JSON й запишіть результати через json.dump() у UTF-8 з ensure_ascii=False та відступом 2.
2. Прочитайте JSON через json.load(). Перевірте наявність ключів, що records є списком, totals — словником, skipped — невід’ємним цілим. Для власного правильно сформованого файлу решта полів відповідає завданню 2.
3. Використовуючи саме прочитані records, заново обчисліть кількість, загальний показник і підсумки груп за своїм варіантом. Порівняйте підсумки груп із прочитаним totals з точністю 10^{-6} та повідомте результат перевірки.

Обробіть OSError читання й запису та JSONDecodeError пошкодженого JSON. Неправильна структура дає повідомлення й припинення обчислень. Перевірте власний результат, порожні records, відсутній файл, пошкоджений JSON і невідповідність збереженого підсумку прочитаним записам. Весь JSON зберігайте в одному файлі; шлях задає користувач.

Для JSON виберіть окремий від CSV файл. Перевірки відсутнього, пошкодженого або зміненого JSON виконуйте окремим запуском читання без попереднього запису. Для перевірки розбіжності змініть у збереженому totals один наявний підсумок на 1, не змінюючи records.

Таблиця 4 — Контрольна перевірка JSON

Дані або дія	Очікуваний результат	Що перевіряється
Зберегти й прочитати результат контрольного CSV.	Кількість, показник і групи відповідають своєму рядку таблиці варіантів; порівняння успішне.	Перерахунок із loaded records
records = [], totals = {}, skipped = 0.	Кількість 0, показник 0.00; порівняння успішне.	Порожні результати
Відсутній файл читання.	Повідомлення про OSError, обчислення не виконуються.	Помилка доступу
Пошкоджений синтаксис JSON.	Повідомлення про JSONDecodeError, обчислення не виконуються.	Синтаксис файла
Немає одного потрібного ключа або records не є списком.	Повідомлення про неправильну структуру; обчислення не виконуються.	Структура даних
У totals збільшено один підсумок на 1; records без змін.	Перерахунок із records дає початкові значення; порівняння з totals неуспішне.	Виявлення зміненого підсумку

Оформлення звіту

Звіт оформлюють за чинними вимогами курсу (документ «Вимоги до оформлення»).

Для кожного завдання у звіті подають формулювання, текст програми та скріншот результату в консолі.

Методичні рекомендації

Текстовий файл, кодування та with

open(path, "r", encoding="utf-8") відкриває файл для читання, "w" — для нового запису з очищенням попереднього вмісту. З with файл закривається і після успіху, і після винятку. Для читання й запису підсумку потрібні різні шляхи, щоб не втратити джерело.

for line in file перебирає фізичні рядки. Порожній рядок між іншими теж рахується. Якщо останній рядок не закінчується символом нового рядка, він усе

одно повертається. Для порожнього файлу цикл не виконується. len(line.split()) рахує слова, відокремлені пробільними символами.

write(text) записує рядок без автоматичного перенесення: додавайте "\n" самостійно. OSError охоплює відсутність файлу, недоступний шлях та інші помилки доступу. Обробляйте читання й запис окремо, щоб повідомлення пояснювало невдалу операцію.

Рядки й слова рахуються по-різному

Код Python

```
with open("демо", "w", encoding="utf-8") as file:
    file.write("Мова Python\n\nКод")
line_count = 0
word_count = 0
with open("демо", encoding="utf-8") as file:
    for line in file:
        line_count += 1
        word_count += len(line.split())
print(line_count, word_count)
```

Результат виконання

```
3 3
```

Демонстрація сама створює короткий файл. Порожній фізичний рядок збільшує line_count, але не word_count. Останній рядок без перенесення теж враховано. У завданні шляхи вводить користувач.

CSV та числові поля

CSV містить текстові поля. import csv підключає модуль. Файл відкривають із newline=""; csv.DictReader(file, delimiter=";") читає записи в словники за заголовком.

csv.DictWriter записує таблиці, writeheader() додає заголовок. Не використовуйте split(";"): роздільник може бути всередині поля в лапках.

DictReader.fieldnames містить назви стовпців. Порівняйте їх зі списком очікуваних назв. Відсутнє поле може дати значення None, зайві поля — ключ None. Перевірте це до strip() і float(). Повністю порожні фізичні рядки читач ігнорує; рядок лише з порожніми полями слід вважати некоректним.

Числові поля залишаються рядками: float("2.5") дає 2.5, float("помилка") піднімає ValueError. Значення NaN та нескінченність теж можуть пройти float(), тому math.isfinite(value) перевіряє скінченність. Для цілої кількості можна

перевірити value.is_integer(). Перевірку додаткових меж свого варіанта виконайте перед формулою.

Некоректний запис і коректний невідібраний запис — різні випадки. Лічильник пропущених змінюйте лише для некоректних. Для сум накопичуйте значення за ключем. Для середніх збережіть також кількість записів кожної групи й поділіть суму на кількість. Для сортування виведення перебирайте sorted(totals); порядок відібраних записів збережіть.

CSV-читач ураховує лапки

Код Python

```
import csv

rows = ["name;score", '"Мова;код";80']
reader = csv.DictReader(rows, delimiter=";", strict=True)
print(reader.fieldnames)
for row in reader:
    print(row["name"], float(row["score"]))
```

Результат виконання

```
['name', 'score']
Мова;код 80.0
```

У назві всередині лапок є «;», але це одне поле. У демонстрації джерело — готові рядки; для реального файла передайте reader відкритий через with файл із newline="". Усі числові поля спочатку є рядками.

Відсутні та зайві поля в записі

Код Python

```
import csv

reader = csv.DictReader(["name;score", "Олена", "Іван;90;зайве"], delimiter=";")
for row in reader:
    invalid = None in row or any(value is None for value in row.values())
    print(invalid)
```

Результат виконання

```
True
True
```

Відсутнє значення дає None, зайві поля потрапляють під ключ None. Цю перевірку потрібно виконати до strip() та числового перетворення.

Формування груп, сума та середнє

У словнику groups кожній назві відповідає список її показників. Спочатку зберемо ці списки з records, потім обчислимо суму й середнє для кожної групи.

Код Python

```
records = [
    {"group": "Група 1", "value": 2.5},
    {"group": "Група 2", "value": 3.0},
    {"group": "Група 1", "value": 3.0},
]
groups = {}
for item in records:
    group = item["group"]
    groups.setdefault(group, []).append(item["value"])
for group in sorted(groups):
    values = groups[group]
    total = sum(values)
    average = total / len(values)
    print(f"{group}: сума {total:.2f}, середнє {average:.2f}")
```

Результат виконання

```
Група 1: сума 5.50, середнє 2.75
Група 2: сума 3.00, середнє 3.00
```

setdefault(group, []) створює порожній список лише для нової групи й повертає список цієї групи. append() додає поточне value. Тому для Група 1 накопичено [2.5, 3.0], а для Група 2 — [3.0]; sorted(groups) задає порядок виведення назв.

Для Група 1 сума — 5.50, середнє — 2.75. У варіанті оберіть потрібний показник: суму або середнє.

Загальне середнє рахуйте з усіх value: 8.5 / 3 ≈ 2.83, а не з групових середніх. Для порожнього набору груп немає, а загальний показник дорівнює 0.

JSON і використання прочитаних результатів

import json підключає серіалізацію. json.dump(data, file, ensure_ascii=False, indent=2) записує структуру, json.load(file) читає її. dump/load працюють із файлами, dumps/loads — із рядками. Список стає JSON-масивом, словник — об’єктом, None — null. Ключі JSON-об’єктів рядкові; множину спочатку потрібно перетворити на список.

ensure_ascii=False зберігає українські літери читабельними. Весь результат записують одним об’єктом: кілька послідовних dump() не утворюють один коректний JSON. JSONDecodeError означає неправильний синтаксис; правильний синтаксис ще не гарантує очікуваної структури.

Після load() перевірте isinstance(data, dict), потрібні ключі та типи полів. Для цілого лічильника type(value) is int відрізняє число від bool. У цій роботі

повний універсальний валідатор сторонніх записів не потрібний: дані records створені вашою програмою за умовою CSV.

Перерахунок виконують із loaded["records"], а не зі старої змінної до запису. Обчисліть суму або середнє за правилом варіанта. Для чисел з рухомою крапкою порівнюйте модуль різниці з допуском, наприклад abs(actual − expected) ≤ 0.000001. Спочатку порівняйте набори ключів груп, потім відповідні значення. Порожні records й totals дають показник 0 та узгоджений результат.

Запис, читання та новий розрахунок

Код Python

```
import json

data = {"values": [2, 5], "total": 7}
with open("структура", "w", encoding="utf-8") as file:
    json.dump(data, file, ensure_ascii=False, indent=2)
with open("структура", encoding="utf-8") as file:
    loaded = json.load(file)
actual = sum(loaded["values"])
print(actual)
print(actual == loaded["total"])
```

Результат виконання

```
7
True
```

load() повернув нову структуру з файла. Розрахунок actual використовує loaded, тому перевіряється збережений результат. У завданні додайте визначену схему та потрібні реакції на помилки.

Межі обробки винятків

try містить операції, які можуть завершитися помилкою, except — реакцію на конкретний тип. Для числового перетворення потрібен вузький try/except ValueError усередині циклу; continue переходить до наступного запису. OSError і csv.Error читання охоплюють зовнішню операцію. Якщо читання обірвалося, не подавайте частковий підсумок як успішний.

Умова може явно підняти ValueError через raise ValueError("опис"), якщо число перетворилося, але порушує межу. Не перехоплюйте всі винятки без уточнення типу: це приховує помилки програми.

Помилка числа не завершує весь цикл

Код Python

```
import math

for field in ["2.5", "помилка", "nan"]:
    try:
        value = float(field)
        if not math.isfinite(value) or value < 0:
            raise ValueError("Недопустиме число")
    except ValueError:
        print(f"Пропущено: {field}")
        continue
    print(f"Значення: {value:.2f}")
```

Результат виконання

```
Значення: 2.50
Пропущено: помилка
Пропущено: nan
```

float() може прийняти nan, тому окрема перевірка скінченності потрібна після перетворення. Вузький except обробляє лише поточне поле; continue переходить до наступного.

Пошкоджений CSV і JSON мають свої винятки

Код Python

```
import csv
import json

try:
    list(csv.reader(['"не закрита'], strict=True))
except csv.Error:
    print("Пошкоджений CSV")
try:
    json.loads("{неправильний JSON}")
except json.JSONDecodeError:
    print("Пошкоджений JSON")
```

Результат виконання

```
Пошкоджений CSV
Пошкоджений JSON
```

strict=True дозволяє читачу відхилити некоректне завершення лапок. JSONDecodeError означає помилку синтаксису. На відміну від некоректного числового поля, такі помилки всього джерела зупиняють відповідну операцію.

Часті помилки

Не відкривайте джерело в режимі "w". Не підміняйте CSV ручним розділенням рядків. Не додавайте до пропущених записи, що просто не

пройшли відбір. Не перевіряйте JSON за змінними до читання. Не округлюйте проміжні суми до завершення обчислень.

Приклади виконання завдань

Приклад до завдання 1

Навчальна задача. Вхідний файл UTF-8 містить по одному слову в непорожньому рядку. Порахуйте слова та загальну кількість їхніх символів, виведіть і збережіть підсумок. Для показаного запуску вміст вхідного файла — «мова», порожній рядок, «Python», «код».

Код Python

```
source_path = input("Вхідний шлях: ")
target_path = input("Вихідний шлях: ")
try:
    count = 0
    letters = 0
    with open(source_path, "r", encoding="utf-8") as file:
        for line in file:
            word = line.strip()
            if word:
                count += 1
                letters += len(word)
except OSError:
    print("Помилка читання файла")
else:
    report = f"Слів: {count}\nСимволів: {letters}\n"
    print(report, end="")
    try:
        with open(target_path, "w", encoding="utf-8") as file:
            file.write(report)
    except OSError:
        print("Не вдалося зберегти підсумок")
    else:
        print("Підсумок збережено")
```

Приклад виконання в консолі

```
Вхідний шлях: джерело
Вихідний шлях: підсумок
Слів: 3
Символів: 13
Підсумок збережено
```

Пояснення до прикладу

Порожній рядок не враховується. Довжини трьох слів — 4, 6 і 3, тому count = 3, letters = 13. Символи нового рядка не входять до підсумку після strip().

Блок else виконується лише після успішного читання. За помилки вихідний файл не відкривається. Окремі try контролюють читання й запис; report забезпечує однаковий підсумок у консолі та файлі.

У завданні 1 враховуйте всі фізичні рядки, включно з порожніми, та рахуйте слова через split(). Вміст створеного файла у цьому прикладі — два рядки «Слів: 3» та «Символів: 13».

Приклад до завдання 2

У CSV з двома полями name;score знайдемо всі оцінки не нижчі за 60 та середнє серед них. Приклад показує читач CSV, пропуск некоректних чисел і відокремлення відбору від помилки.

Перед запуском створіть вхідний файл оцінки з таким вмістом.

```
name;score
Олена;80
Іван;70
Анна;50
Богдан;помилка
```

Код Python

```
import csv
import math

path = input("Шлях CSV: ")
selected = []
skipped = 0
try:
    with open(path, encoding="utf-8", newline="") as file:
        reader = csv.DictReader(file, delimiter=";", strict=True)
        if reader.fieldnames != ["name", "score"]:
            raise ValueError("Неправильний заголовок")
        for row in reader:
            try:
                if None in row or any(v is None for v in row.values()):
                    raise ValueError("Неправильна кількість полів")
                name = row["name"].strip()
                score = float(row["score"].strip())
                if not name or not math.isfinite(score) or not 0 <= score <= 100:
                    raise ValueError("Недопустимі поля")
            except ValueError:
                skipped += 1
                print("Некоректний запис пропущено")
                continue
            if score >= 60:
                selected.append({"name": name, "score": score})
except (OSError, csv.Error, ValueError) as error:
    print(f"Читання не завершено: {error}")
else:
    for item in selected:
        print(f"{item['name']}: {item['score']:.2f}")
    average = sum(item["score"] for item in selected) / len(selected) if selected else 0
    print(f"Відібрано: {len(selected)}; середнє: {average:.2f}")
    print(f"Некоректних: {skipped}")
```

Приклад виконання в консолі

```
Шлях CSV: оцінки
Некоректний запис пропущено
Олена: 80.00
Іван: 70.00
Відібрано: 2; середнє: 75.00
Некоректних: 1
```

Пояснення до прикладу

DictReader перетворює запис на словник із ключами із заголовка. Перед обчисленнями перевіряються кількість полів і назва, а score перетворюється на число. isfinite() відхиляє nan та нескінченність; далі перевіряється межа від 0 до 100 включно.

Внутрішній except ValueError обробляє один запис: збільшує skipped і виконує continue. Бали 50 допустимі, але не проходять відбір; «помилка» — некоректне число. Тому відібрано два записи, а некоректний лише один.

Зовнішній try охоплює відкриття й читання CSV. Його else виводить підсумок лише після успішного завершення: (80 + 70) / 2 = 75.00. Для порожнього selected умовний вираз повертає 0 без ділення на нуль.

Приклад до завдання 3

Збережіть опитування дією «запис», а потім окремим запуском виконайте «читання». За рівності голосів перемагає перша назва за sorted(); для порожнього опитування виводимо «немає».

Код Python

```
import json

path = input("Шлях JSON: ")
action = input("Дія (запис/читання): ").strip().lower()
if action == "запис":
    votes = {"ранок": 3, "вечір": 5}
    data = {"votes": votes, "total": sum(votes.values())}
    try:
        with open(path, "w", encoding="utf-8") as file:
            json.dump(data, file, ensure_ascii=False, indent=2)
    except OSError as error:
        print(f"Не вдалося записати: {error}")
    else:
        print("Опитування збережено")
elif action == "читання":
    try:
        with open(path, encoding="utf-8") as file:
            loaded = json.load(file)
        if not isinstance(loaded, dict) or not isinstance(loaded.get("votes"), dict):
            raise ValueError("Неправильна структура")
        if type(loaded.get("total")) is not int:
            raise ValueError("Неправильний підсумок")
        if any(type(v) is not int or v < 0 for v in loaded["votes"].values()):
            raise ValueError("Неправильні голоси")
        actual = sum(loaded["votes"].values())
        winner = None
        for name in sorted(loaded["votes"]):
            if winner is None or loaded["votes"][name] > loaded["votes"][winner]:
                winner = name
        print(f"Голосів після читання: {actual}")
        print(f"Перевірка: {actual == loaded['total']}")
        print(f"Переможець: {winner if winner is not None else 'немає'}")
    except (OSError, json.JSONDecodeError, ValueError) as error:
        print(f"Не вдалося прочитати результат: {error}")
else:
    print("Невідома дія")
```

Приклад виконання в консолі

```
Шлях JSON: опитування
Дія (запис/читання): запис
Опитування збережено

Шлях JSON: опитування
Дія (запис/читання): читання
Голосів після читання: 8
Перевірка: True
Переможець: вечір
```

Пояснення до прикладу

У консолі показано два окремі запуски тієї самої програми. Під час першого дія «запис» створює один JSON-об’єкт із votes і total. Під час другого дія «читання» відкриває готовий файл і нічого в нього не записує.

load() повертає нову структуру loaded. До обчислень перевіряються словники, цілий підсумок і невід’ємні цілі голоси. Перевірка type(v) is int також відхиляє True і False. actual = 3 + 5 = 8 обчислюється саме з loaded; у «вечір» 5 голосів, тому це переможець.

Щоб перевірити розбіжність, після запису змініть total у файлі з 8 на 9 та знову оберіть «читання»: сума залишиться 8, а перевірка дасть False. Окремо спробуйте відсутній файл і файл із пошкодженим JSON. Помилки доступу, синтаксису та структури обробляються до виведення результатів.

Контрольні питання

1. Чим відрізняються режими "r" і "w"?
2. Для чого потрібні with та encoding?
3. Як врахувати порожні фізичні рядки?
4. Чому CSV не слід читати через звичайний split()?
5. Що роблять DictReader, DictWriter і writeheader()?
6. Як відрізняються некоректний та невідібраний записи?
7. Чому після float() може знадобитися isfinite()?
8. Чим dump/load відрізняються від dumps/loads?
9. Що перевіряти після успішного читання JSON?
10. Навіщо перераховувати показники з прочитаних даних?