Лабораторна робота №2

Робота з файлами та виведення статистики в Python

Мета: ознайомитись з методами для роботи з файлами в середовищі Google Colab. Набуття базових навичок читання, створення, записування та обробки файлів різних форматів. Закріплення навичок виведення статистичних даних.

2.1 Теоретичні відомості

Файл можна розглядати як контейнер для збереження інформації. Щоб отримати доступ до його вмісту або записати нові дані, його спершу потрібно відкрити. Коли робота з файлом завершена — читання чи запис — його необхідно закрити, аби звільнити використані системні ресурси.

Отже, робота з файлами в Python зазвичай відбувається у такій послідовності:

1. відкриття файлу;
2. виконання читання або запису;
3. закриття файлу.

Відкрити файл можна за допомогою функції open():

```
file = open("lab2/text.txt")
```

За замовчуванням файли відкриті в **режимі читання** (не можуть бути змінені).

```
file = open("text.txt")      # рівнозначний 'r' або 'rt'
file = open("text.txt",'w')  # запис у текстовому режимі
file = open("img.bmp",'r+b')
```

Таблиця 2.1 – Основні режими роботи з файлами.

Режим	Опис
r	Відкриває файл тільки для читання. Якщо файл не існує — виникає помилка. (режим за замовчуванням)
w	Відкриває файл для запису. Якщо файл існує — його вміст стирається. Якщо не існує — створюється новий.
x	Створює новий файл і відкриває його для запису. Якщо файл уже існує — виникає помилка.
a	Відкриває файл для запису в кінець. Поточний вміст зберігається. Якщо файл не існує — він створюється.
t	Відкриває файл у текстовому режимі (стандартний варіант).
b	Відкриває файл у двійковому режимі (наприклад, для зображень чи аудіо).
+	Дозволяє одночасно читати й записувати у файл.

В Python для читання файлів використовується **метод read()**. Наприклад:

```
file = open("lab2/server_logs.txt")
content = file.read()
print(content)
```

Коли робота з файлом завершена, його необхідно закрити. Це дозволяє звільнити ресурси, які використовувались під час доступу до файлу. Закриття виконується за допомогою методу close().

```
file = open("lab2/server_logs.txt")
```

```
content = file.read()
print(content)
file.close()
```

У цьому прикладі ми застосували метод close() для закриття файлу. Завжди після виконання операцій із файлом потрібно його закривати — це є важливим кроком. Якщо під час роботи з файлом стається помилка, програма може завершитися, не закривши файл. Щоб цього уникнути, можна використати конструкцію try...finally, яка гарантує закриття файлу незалежно від того, виникла помилка чи ні.

```
try:
  file = open("lab2/server_logs.txt")
  content = file.read()
  print(content)
finally:
  file.close()
```

Тут ми закрили файл у блоці finally, оскільки блок finally завжди виконується, то файл буде закритий, навіть якщо згенерується виняток.

У Python можна застосовувати конструкцію with...open, яка автоматично закриває файл після завершення роботи з ним.

```
with open("lab2/server_logs.txt", "r") as f:
  content = f.read()
  print(content)
```

Таблиця 2.2 – Методи для роботи з файлами

Метод	Опис
close()	Закриває відкритий файл. Якщо файл уже закритий — нічого не відбувається.
detach()	Від’єднує базовий двійковий буфер від TextIOBase і повертає його.
fileno()	Повертає ціле число — файловий дескриптор.
flush()	Очищає буфер запису файлового потоку, записуючи дані на диск.
isatty()	Повертає True, якщо потік є інтерактивним (наприклад, консоль).
read(n)	Зчитує до n символів з файлу. Якщо n від’ємне або None — читає до кінця файлу.
readable()	Повертає True, якщо з файлу можна зчитувати дані.
readline(n=-1)	Зчитує один рядок з файлу. Можна обмежити кількість символів n.
readlines(n=-1)	Зчитує всі рядки з файлу у список. Можна обмежити загальну кількість символів/байтів n.
seek(offset, from=SEEK_SET)	Змінює поточну позицію у файлі на offset байтів відносно from (початок, поточна позиція, кінець).
seekable()	Повертає True, якщо файл підтримує довільний доступ.
tell()	Повертає поточну позицію у файлі у вигляді числа байтів від початку.
truncate(size=None)	Змінює розмір файлу до size байтів. Якщо size не вказано — до поточної позиції.
writable()	Повертає True, якщо у файл можна записувати дані.
write(s)	Записує рядок s у файл і повертає кількість записаних символів.
writelines(lines)	Записує список рядків lines у файл без додаткових роздільників рядків.

CSV (comma-separated value) - це формат подання табличних даних (наприклад, це можуть бути дані з таблиці або дані з БД).

У цьому форматі кожен рядок файлу – це рядок таблиці. Незважаючи на назву формату, роздільником може бути не тільки кома.

І хоча у форматів з іншим роздільником може бути і власна назва, наприклад, TSV (tab separated values), проте під форматом CSV розуміють, як правило, будь-які роздільники.

Читання здійснюється за допомогою конструкцій:

```
import csv
with open('data.csv', "r") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

У стандартній бібліотеці Python є модуль CSV, який дозволяє працювати з файлами в CSV форматі. У першому списку є назви стовпців, а інших відповідні значення.

Зауважте, що сам csv.reader повертає ітератор. При необхідності його можна перетворити на список таким чином:

```
import csv
with open('lab2/data.csv', "r") as f:
    reader = csv.reader(f)
    print(list(reader))
```

Найчастіше заголовки стовпців зручніше отримати окремим об'єктом. Це можна зробити таким чином:

```
import csv


with open('lab2/data.csv', "r") as f:
    reader = csv.reader(f)
    headers = next(reader)
    print('Headers: ', headers)
```

Іноді в результаті обробки набагато зручніше отримати словники, в яких ключі – це назви стовпців, а значення – значення стовпців. Для цього в модулі є DictReader:

```
import csv
with open('lab2/data.csv', "r") as f:
    reader = csv.DictReader(f)
    for row in reader:
      print(row["Date"], row["Temperature"])
```

Аналогічно за допомогою модуля csv можна і записати файл у форматі CSV

```
data = [
    ['Date', 'SensorID', 'Temperature', 'Humidity', 'Pressure', 'Status', 'Active'],
    ['2025-09-01', 'S001', 22.5, 45, 1012, 'OK', True],
    ['2025-09-01', 'S002', 23.0, 47, 1011, 'OK', True],
    ['2025-09-01', 'S003', 21.8, 44, 1013, 'FAULT', False],
    ['2025-09-02', 'S001', 22.2, 46, 1012, 'OK', True]]
with open('data.csv', 'w') as f:
    writer = csv.writer(f)
    for row in data:
        writer.writerow(row)
```

```
with open('data.csv') as f:
    print(f.read())
```

Для того, щоб усі рядки записувалися в CSV-файл із лапками, треба змінити скрипт таким чином:

```
data = [
    ['Date', 'SensorID', 'Temperature', 'Humidity', 'Pressure', 'Status', 'Active'],
    ['2025-09-01', 'S001', 22.5, 45, 1012, 'OK', True],
    ['2025-09-01', 'S002', 23.0, 47, 1011, 'OK', True],
    ['2025-09-01', 'S003', 21.8, 44, 1013, 'FAULT', False],
    ['2025-09-02', 'S001', 22.2, 46, 1012, 'OK', True]]
with open('data.csv', 'w') as f:
    writer = csv.writer(f, quoting=csv.QUOTE_NONNUMERIC)
    for row in data:
        writer.writerow(row)

with open('data.csv') as f:
    print(f.read())
```

2.2 Порядок виконання лабораторної роботи

1. Ознайомитись з теоретичними відомостями.

2. Створити файл server_logs.txt та записати логи сервера в додатку 1;

3. На основі файлу server_logs.txt:

- Підрахувати кількість унікальних IP;
- Визначити найчастіші HTTP-методи;
- Знайти кількість помилок;
- Порахувати загальний обсяг переданих даних;
- Відфільтрувати всі запити “api”;
- Записати дані у новий файл results.txt та побудувати графіки найчастіших HTTP-методів.

4. Створити файл data.csv записати в нього дані з додатку 2;

5. На основі файлу data.csv виконати:

- Читання даних з файлу;
- Підрахувати кількість рядків та вивести кількість активних сенсорів;
- Відфільтрувати данні в яких Status=FAULT, в новий файл записати активні датчики Active=True;
- Створити список даних для S001 датчика;
- Обчислити середню температуру для кожного сенсора окремо;
- Знайти мінімальне та максимальне значення вологості по всім даним;
- Побудувати графік вологості з вказанням найбільшого та найменшого значення.

2.3 Зміст звіту

1. Найменування і мета роботи;
2. Код по кожному пункту порядку виконання роботи;
3. Результати роботи по кожному пункту виконання роботи;
4. Висновки.

2.4 Контрольні запитання

1. Які типи файлів ви можете відкривати у Python?
2. Що таке режим відкриття файлу (r, w, a, rb, wb)?
3. Чим відрізняється текстовий файл від бінарного у Python?
4. Що відбувається, якщо забути закрити файл?
5. Як прочитати весь текст із файлу example.txt за один раз?
6. Як прочитати файл рядок за рядком?
7. Як додати новий рядок у кінець файлу?
8. Як фільтрувати рядки CSV за значенням у конкретному стовпці?
9. Як записати нові дані у CSV з додаванням нового стовпця?
10. Як працювати з числами та текстом у CSV-файлі одночасно?

Додаток 1

server_log.txt

192.168.1.10 - - [11/Sep/2025:12:34:56 +0300] "GET /index.html HTTP/1.1" 200 5320
192.168.1.11 - - [11/Sep/2025:12:35:02 +0300] "POST /login HTTP/1.1" 302 1024
192.168.1.15 - - [11/Sep/2025:12:35:10 +0300] "GET /dashboard HTTP/1.1" 200 8743
192.168.1.12 - - [11/Sep/2025:12:35:20 +0300] "GET /images/logo.png HTTP/1.1" 404 512
192.168.1.14 - - [11/Sep/2025:12:35:33 +0300] "GET /api/data HTTP/1.1" 500 2048
192.168.1.10 - - [11/Sep/2025:12:35:40 +0300] "GET /contact HTTP/1.1" 200 3100
192.168.1.13 - - [11/Sep/2025:12:35:55 +0300] "PUT /api/update HTTP/1.1" 201 1456
192.168.1.16 - - [11/Sep/2025:12:36:01 +0300] "DELETE /api/remove HTTP/1.1" 403 678
192.168.1.11 - - [11/Sep/2025:12:36:20 +0300] "GET /index.html HTTP/1.1" 200 5320
192.168.1.15 - - [11/Sep/2025:12:36:44 +0300] "GET /unknown HTTP/1.1" 404 256

Додаток 2

data.csv

Date,SensorID,Temperature,Humidity,Pressure,Status,Active
2025-09-01,S001,22.5,45,1012,OK,True
2025-09-01,S002,23.0,47,1011,OK,True
2025-09-01,S003,21.8,44,1013,FAULT,False
2025-09-02,S001,22.2,46,1012,OK,True
2025-09-02,S002,23.5,48,1010,OK,True
2025-09-02,S003,24.0,49,1011,OK,True
2025-09-03,S001,22.8,45,1012,OK,True
2025-09-03,S002,23.2,46,1013,OK,True
2025-09-03,S003,22.0,44,1012,FAULT,False
2025-09-04,S001,23.5,47,1010,OK,True
2025-09-04,S002,24.0,49,1011,OK,True
2025-09-04,S003,22.5,45,1012,OK,True
2025-09-05,S001,23.8,48,1013,OK,True
2025-09-05,S002,22.9,46,1010,OK,True
2025-09-05,S003,23.1,47,1011,FAULT,False
2025-09-06,S001,22.7,45,1012,OK,True
2025-09-06,S002,23.4,48,1013,OK,True
2025-09-06,S003,24.0,49,1011,OK,True
2025-09-07,S001,22.9,46,1010,OK,True
2025-09-07,S002,23.6,47,1012,OK,True
2025-09-07,S003,22.8,45,1013,FAULT,False