Python Programming - File Handling (Xử Lý File)

> **Mục tiêu học tập**: Làm chủ đọc/ghi file trong Python - kỹ năng thiết yếu cho các bài Assessment và ứng dụng thực tế.

1. Giới Thiệu File Handling

1.1. Tại Sao Cần File Handling?

🤔 Đặt vấn đề

Dữ liệu trong chương trình bị mất khi tắt máy. Làm sao để lưu trữ lâu dài?

```
# ❌ Dữ liệu chỉ tồn tại trong runtime
students = ["An", "Bình", "Chi"]
# Tắt chương trình → mất hết!
```

💡 Giải quyết: Files

File cho phép lưu trữ dữ liệu **persistent** (vĩnh viễn) trên ổ cứng.

```
# ☑ Ghi vào file
with open("students.txt", "w") as f:
    for student in students:
        f.write(student + "\n")

# ☑ Đọc từ file (sau khi restart)
with open("students.txt", "r") as f:
    students = [line.strip() for line in f]
    print(students)  # ['An', 'Bình', 'Chi']
```

1.2. File Operations

4 thao tác cơ bản:

1. **Open** - Mở file
2. **Read/Write** - Đọc/ghi dữ liệu
3. **Close** - Đóng file (giải phóng tài nguyên)
4. **Process** - Xử lý dữ liệu

2. Mở File với `open()`

2.1. Cú Pháp

```
file_object = open(filename, mode, encoding)
```

Parameters:

- `filename` (str): Đường dẫn đến file
- `mode` (str): Chế độ mở file
- `encoding` (str): Encoding (khuyến nghị `utf-8`)

2.2. File Modes

Mode	Ý nghĩa	Tạo mới nếu không tồn tại	Lỗi nếu không tồn tại
`'r'`	**Read** (đọc)	❌	☑ FileNotFoundError
`'w'`	**Write** (ghi, xóa nội dung cũ)	☑	❌
`'a'`	**Append** (ghi tiếp vào cuối)	☑	❌
`'r+'`	Read + Write	❌	☑
`'w+'`	Write + Read (xóa cũ)	☑	❌
`'a+'`	Append + Read	☑	❌

Text vs Binary:

- `'r'`, `'w'`, `'a'` - **Text mode** (mặc định)
- `'rb'`, `'wb'`, `'ab'` - **Binary mode** (cho file ảnh, video, etc.)

2.3. Ví Dụ Mở File

```
# Mode 'r' - Đọc
f = open("data.txt", "r")  # Lỗi nếu file không tồn tại
content = f.read()
f.close()

# Mode 'w' - Ghi (xóa nội dung cũ)
f = open("output.txt", "w")
f.write("Hello World")
f.close()

# Mode 'a' - Ghi tiếp (không xóa)
f = open("log.txt", "a")
f.write("New log entry\n")
f.close()

# Với encoding (QUAN TRỌNG cho tiếng Việt)
f = open("data.txt", "r", encoding="utf-8")
content = f.read()
f.close()
```

2.4. ⚠ Luôn Đóng File

```
# ❌ Quên close() - tốn tài nguyên, có thể mất dữ liệu
f = open("data.txt", "r")
content = f.read()
# Quên f.close()!

# ☑ Nhớ close()
f = open("data.txt", "r")
content = f.read()
f.close()  # Giải phóng tài nguyên

# ☑☑ BEST: Dùng context manager (tự động close)
with open("data.txt", "r") as f:
    content = f.read()
# File tự động close khi ra khỏi block
```

---

3. Đọc File

3.1. `.read()` - Đọc Toàn Bộ

```
# Đọc toàn bộ file thành 1 string
with open("data.txt", "r", encoding="utf-8") as f:
    content = f.read()
    print(content)

# Đọc n ký tự đầu
with open("data.txt", "r") as f:
    first_100_chars = f.read(100)
    print(first_100_chars)
```

Ví dụ: Đếm số từ trong file

```
def count_words(filename):
    """Đếm số từ trong file"""
    with open(filename, "r", encoding="utf-8") as f:
        content = f.read()
        words = content.split()
        return len(words)

# Test
print(f"Số từ: {count_words('data.txt')}")
```

3.2. `.readline()` - Đọc Từng Dòng

```
# Đọc 1 dòng
with open("data.txt", "r") as f:
    line1 = f.readline()  # Dòng 1
    line2 = f.readline()  # Dòng 2
    print(line1)
    print(line2)

# Đọc tất cả dòng với vòng lặp
with open("data.txt", "r") as f:
    line = f.readline()
    while line:
        print(line.strip())  # strip() xóa \n cuối dòng
        line = f.readline()
```

3.3. `.readlines()` - Đọc Thành List

```
# Đọc tất cả dòng thành list
with open("data.txt", "r", encoding="utf-8") as f:
    lines = f.readlines()
    # lines = ['Line 1\n', 'Line 2\n', 'Line 3\n']

    for line in lines:
        print(line.strip())

# Xóa \n ngay khi đọc
with open("data.txt", "r") as f:
    lines = [line.strip() for line in f.readlines()]
    print(lines)  # ['Line 1', 'Line 2', 'Line 3']
```

3.4. Duyệt Trực Tiếp File Object

```
# ☑ BEST - Pythonic và tiết kiệm bộ nhớ
with open("data.txt", "r", encoding="utf-8") as f:
    for line in f:  # File object là iterable
        print(line.strip())

# Với enumerate để có số dòng
with open("data.txt", "r") as f:
    for i, line in enumerate(f, 1):
        print(f"Dòng {i}: {line.strip()}")
```

3.5. So Sánh Các Cách Đọc

Method	Return	Use Case
`.read()`	String (toàn bộ)	File nhỏ, cần xử lý toàn văn bản

Method	Return	Use Case
`.readline()`	String (1 dòng)	Đọc từng dòng một
`.readlines()`	List of strings	Cần random access các dòng
`for line in f:`	Iterate từng dòng	**BEST** - file lớn, xử lý tuần tự

Ví dụ: Đọc file scores (Bài 3 Assessment)

```
def read_scores(filename):
    """
    Đọc file scores.txt
    Format: student_id,subject,score

    Returns:
        list: [(student_id, subject, score), ...]
    """
    scores = []

    with open(filename, "r", encoding="utf-8") as f:
        for line in f:
            line = line.strip()
            if line:  # Bỏ dòng trống
                parts = line.split(',')
                student_id = parts[0]
                subject = parts[1]
                score = int(parts[2])
                scores.append((student_id, subject, score))

    return scores

# Test
scores = read_scores("scores.txt")
for student_id, subject, score in scores:
    print(f"{student_id} - {subject}: {score}")
```

---

4. Ghi File

4.1. `.write()` - Ghi String

```
# Ghi 1 string
with open("output.txt", "w", encoding="utf-8") as f:
    f.write("Hello World\n")
    f.write("Python is awesome\n")

# ⚠ write() KHÔNG tự xuống dòng
with open("output.txt", "w") as f:
    f.write("Line 1")
```

```
    f.write("Line 2")  # Nối liền: "Line 1Line 2"

# ☑ Phải thêm \n
with open("output.txt", "w") as f:
    f.write("Line 1\n")
    f.write("Line 2\n")
```

4.2. `.writelines()` - Ghi List

```
# Ghi list strings
lines = ["Line 1\n", "Line 2\n", "Line 3\n"]

with open("output.txt", "w") as f:
    f.writelines(lines)

# ⚠ writelines() KHÔNG tự thêm \n
lines = ["Line 1", "Line 2", "Line 3"]

with open("output.txt", "w") as f:
    f.writelines(lines)  # Nối liền: "Line 1Line 2Line 3"

# ☑ Thêm \n vào mỗi line
with open("output.txt", "w") as f:
    for line in lines:
        f.write(line + "\n")

# Hoặc dùng join
with open("output.txt", "w") as f:
    f.write("\n".join(lines) + "\n")
```

4.3. Mode 'a' - Append

```
# Mode 'w' - XÓA nội dung cũ
with open("log.txt", "w") as f:
    f.write("Entry 1\n")

with open("log.txt", "w") as f:
    f.write("Entry 2\n")  # File chỉ có "Entry 2"

# Mode 'a' - GHI TIẾP vào cuối
with open("log.txt", "a") as f:
    f.write("Entry 1\n")

with open("log.txt", "a") as f:
    f.write("Entry 2\n")  # File có cả 2 entries
```

Ví dụ: Ghi kết quả ra file

```python
def save_results(filename, students):
    """
    Ghi kết quả ra file

    Args:
        students (list): [(name, score), ...]
    """
    with open(filename, "w", encoding="utf-8") as f:
        f.write("=== BẢNG ĐIỂM ===\n")
        f.write("-" * 30 + "\n")

        for name, score in students:
            f.write(f"{name}: {score}\n")

        f.write("-" * 30 + "\n")

# Test
students = [("An", 8.5), ("Bình", 9.0), ("Chi", 7.5)]
save_results("results.txt", students)
```

---

5. Context Manager: `with` Statement

5.1. Tại Sao Cần `with`?

❌ Vấn đề: Quên close()

```python
# Nếu có lỗi → file không được close
f = open("data.txt", "r")
data = f.read()
result = process(data)  # Lỗi ở đây!
f.close()  # Không chạy đến dòng này!
```

❌ Vấn đề: Phải try-finally

```python
# Phải dùng try-finally
f = open("data.txt", "r")
try:
    data = f.read()
    result = process(data)
finally:
    f.close()  # Luôn chạy
```

☑ Giải quyết: `with` statement

```
# Tự động close, ngay cả khi có lỗi
with open("data.txt", "r") as f:
    data = f.read()
    result = process(data)  # Có lỗi cũng OK
# File tự động close ở đây!
```

5.2. Cú Pháp `with`

```
with open(filename, mode) as file_object:
    # Code xử lý file
    # ...
# File tự động close
```

5.3. Nhiều Files Cùng Lúc

```
# Đọc từ file này, ghi sang file kia
with open("input.txt", "r") as fin, open("output.txt", "w") as fout:
    for line in fin:
        fout.write(line.upper())
```

5.4. Best Practices

```
# ☑ LUÔN dùng with
with open("data.txt", "r", encoding="utf-8") as f:
    content = f.read()

# ☑ LUÔN chỉ định encoding
with open("data.txt", "r", encoding="utf-8") as f:
    content = f.read()

# ☑ Xử lý FileNotFoundError
try:
    with open("data.txt", "r") as f:
        content = f.read()
except FileNotFoundError:
    print("File không tồn tại!")
```

---

6. Xử Lý CSV Files

6.1. CSV là gì?

CSV (Comma-Separated Values) là format file text lưu dữ liệu dạng bảng.

```
name,age,city
An,25,Hà Nội
Bình,30,TP.HCM
Chi,22,Đà Nẵng
```

6.2. Đọc CSV Thủ Công (Simple)

```python
def read_csv_manual(filename):
    """
    Đọc CSV thủ công với split()

    Returns:
        list: [["An", "25", "Hà Nội"], ...]
    """
    rows = []

    with open(filename, "r", encoding="utf-8") as f:
        for line in f:
            line = line.strip()
            if line:
                parts = line.split(',')
                rows.append(parts)

    return rows

# Test
data = read_csv_manual("students.csv")
for row in data:
    print(row)
# ['name', 'age', 'city']
# ['An', '25', 'Hà Nội']
# ['Bình', '30', 'TP.HCM']
```

Xử lý header:

```python
def read_csv_with_header(filename):
    """Đọc CSV, tách header và data"""
    with open(filename, "r", encoding="utf-8") as f:
        lines = [line.strip() for line in f if line.strip()]

    if not lines:
        return [], []

    header = lines[0].split(',')
    data = [line.split(',') for line in lines[1:]]

    return header, data
```

```
# Test
header, data = read_csv_with_header("students.csv")
print(f"Header: {header}")  # ['name', 'age', 'city']
print(f"Data: {data}")       # [['An', '25', 'Hà Nội'], ...]
```

6.3. Module `csv` - csv.reader

```
import csv

# Đọc CSV với csv.reader
with open("students.csv", "r", encoding="utf-8") as f:
    reader = csv.reader(f)

    for row in reader:
        print(row)  # List: ['An', '25', 'Hà Nội']

# Tách header
with open("students.csv", "r", encoding="utf-8") as f:
    reader = csv.reader(f)
    header = next(reader)  # Đọc dòng đầu

    for row in reader:
        name, age, city = row
        print(f"{name} - {age} tuổi - {city}")
```

6.4. Module `csv` - csv.DictReader

```
import csv

# Đọc CSV thành dictionaries (KHUYẾN NGHỊ)
with open("students.csv", "r", encoding="utf-8") as f:
    reader = csv.DictReader(f)

    for row in reader:
        # row là dict: {'name': 'An', 'age': '25', 'city': 'Hà Nội'}
        print(f"{row['name']} - {row['age']} tuổi - {row['city']}")

# Chuyển thành list of dicts
with open("students.csv", "r", encoding="utf-8") as f:
    reader = csv.DictReader(f)
    students = list(reader)

print(students)
# [
#   {'name': 'An', 'age': '25', 'city': 'Hà Nội'},
#   {'name': 'Bình', 'age': '30', 'city': 'TP.HCM'},
#   ...
# ]
```

6.5. Ghi CSV

Ghi thủ công

```
def write_csv_manual(filename, data):
    """Ghi CSV thủ công"""
    with open(filename, "w", encoding="utf-8") as f:
        for row in data:
            line = ",".join(str(item) for item in row)
            f.write(line + "\n")

# Test
data = [
    ["name", "age", "city"],
    ["An", 25, "Hà Nội"],
    ["Bình", 30, "TP.HCM"]
]

write_csv_manual("output.csv", data)
```

Ghi với csv.writer

```
import csv

data = [
    ["name", "age", "city"],
    ["An", 25, "Hà Nội"],
    ["Bình", 30, "TP.HCM"]
]

with open("output.csv", "w", encoding="utf-8", newline='') as f:
    writer = csv.writer(f)
    writer.writerows(data)  # Ghi tất cả rows

    # Hoặc ghi từng row
    # writer.writerow(["name", "age", "city"])
    # writer.writerow(["An", 25, "Hà Nội"])
```

Ghi với csv.DictWriter

```
import csv

students = [
    {"name": "An", "age": 25, "city": "Hà Nội"},
    {"name": "Bình", "age": 30, "city": "TP.HCM"}
]
```

```
with open("output.csv", "w", encoding="utf-8", newline='') as f:
    fieldnames = ["name", "age", "city"]
    writer = csv.DictWriter(f, fieldnames=fieldnames)

    writer.writeheader()  # Ghi header
    writer.writerows(students)  # Ghi data
```

---

7. Ứng Dụng Cho Assessment

7.1. Bài 2: Đọc File Strings

```
def read_strings_file(filename):
    """
    Đọc file chứa strings (mỗi dòng 1 string)

    Returns:
        list: Danh sách strings
    """
    strings = []

    with open(filename, "r", encoding="utf-8") as f:
        for line in f:
            line = line.strip()
            if line:  # Bỏ dòng trống
                strings.append(line)

    return strings

# Hoặc ngắn gọn hơn
def read_strings_file_v2(filename):
    with open(filename, "r", encoding="utf-8") as f:
        return [line.strip() for line in f if line.strip()]

# Test
strings = read_strings_file("strings.txt")
print(strings)  # ['hello', 'world', 'python', ...]
```

7.2. Bài 3: Đọc CSV Scores

```
def read_scores_csv(filename):
    """
    Đọc file CSV scores
    Format: student_id,subject,score

    Returns:
        dict: {student_id: {subject: score}}
    """
    scores = {}
```

```
    with open(filename, "r", encoding="utf-8") as f:
        # Bỏ header (nếu có)
        next(f, None)  # Skip dòng đầu

        for line in f:
            line = line.strip()
            if not line:
                continue

            parts = line.split(',')
            student_id = parts[0].strip()
            subject = parts[1].strip()
            score = int(parts[2].strip())

            # Tạo dict cho student nếu chưa có
            if student_id not in scores:
                scores[student_id] = {}

            scores[student_id][subject] = score

    return scores

# Test
scores = read_scores_csv("scores.csv")
print(scores)
# {
#   'S001': {'Math': 85, 'Physics': 92, 'Chemistry': 78},
#   'S002': {'Math': 88, 'Physics': 84, 'Chemistry': 90}
# }
```

Với csv module:

```
import csv

def read_scores_csv_v2(filename):
    """Đọc scores với csv.DictReader"""
    scores = {}

    with open(filename, "r", encoding="utf-8") as f:
        reader = csv.DictReader(f)

        for row in reader:
            student_id = row['student_id']
            subject = row['subject']
            score = int(row['score'])

            if student_id not in scores:
                scores[student_id] = {}

            scores[student_id][subject] = score
```

```
    return scores
```

7.3. Bài 5: Đọc File Codes

```
def read_codes_file(filename):
    """
    Đọc file chứa codes (mỗi dòng 1 code)

    Returns:
        list: Danh sách codes
    """
    codes = []

    with open(filename, "r", encoding="utf-8") as f:
        for line in f:
            code = line.strip()
            if code:
                codes.append(code)

    return codes

# Test
codes = read_codes_file("codes.txt")
print(codes)  # ['ABC123', 'DEF456', 'GHI789', ...]
```

---

8. Error Handling

8.1. FileNotFoundError

```
def safe_read_file(filename):
    """Đọc file an toàn"""
    try:
        with open(filename, "r", encoding="utf-8") as f:
            return f.read()
    except FileNotFoundError:
        print(f"Lỗi: File '{filename}' không tồn tại!")
        return None
    except PermissionError:
        print(f"Lỗi: Không có quyền đọc file '{filename}'!")
        return None
    except Exception as e:
        print(f"Lỗi không xác định: {e}")
        return None

# Test
content = safe_read_file("data.txt")
```

```
    if content:
        print(content)
```

8.2. UnicodeDecodeError

```
def read_with_encoding(filename):
    """Thử nhiều encoding"""
    encodings = ['utf-8', 'latin-1', 'cp1252']

    for encoding in encodings:
        try:
            with open(filename, "r", encoding=encoding) as f:
                return f.read()
        except UnicodeDecodeError:
            continue

    print("Không thể đọc file với các encoding thử nghiệm!")
    return None
```

8.3. Validate File Exists

```
import os

def read_if_exists(filename):
    """Kiểm tra file tồn tại trước khi đọc"""
    if not os.path.exists(filename):
        print(f"File '{filename}' không tồn tại!")
        return None

    if not os.path.isfile(filename):
        print(f"'{filename}' không phải file!")
        return None

    with open(filename, "r", encoding="utf-8") as f:
        return f.read()
```

---

9. Bài Tập Thực Hành

Bài 1: Đọc và đếm số dòng

```
def count_lines(filename):
    """
    Đếm số dòng trong file (không tính dòng trống)

    Returns:
```

```
        int: Số dòng
    """
    count = 0

    with open(filename, "r", encoding="utf-8") as f:
        for line in f:
            if line.strip():  # Bỏ dòng trống
                count += 1

    return count

# Test
print(f"Số dòng: {count_lines('data.txt')}")
```

Bài 2: Copy file

```
def copy_file(source, destination):
    """Copy nội dung từ source sang destination"""
    with open(source, "r", encoding="utf-8") as fin:
        content = fin.read()

    with open(destination, "w", encoding="utf-8") as fout:
        fout.write(content)

    print(f"Đã copy từ '{source}' sang '{destination}'")

# Test
copy_file("input.txt", "output.txt")
```

Bài 3: Merge files

```
def merge_files(input_files, output_file):
    """
    Merge nhiều files thành 1

    Args:
        input_files (list): Danh sách file cần merge
        output_file (str): File đích
    """
    with open(output_file, "w", encoding="utf-8") as fout:
        for filename in input_files:
            with open(filename, "r", encoding="utf-8") as fin:
                fout.write(f"\n=== {filename} ===\n")
                fout.write(fin.read())
                fout.write("\n")

    print(f"Đã merge {len(input_files)} files vào '{output_file}'")
```

```
# Test
merge_files(["file1.txt", "file2.txt", "file3.txt"], "merged.txt")
```

Bài 4: Filter lines

```
def filter_lines(input_file, output_file, keyword):
    """
    Lọc các dòng chứa keyword

    Args:
        keyword (str): Từ khóa cần tìm
    """
    count = 0

    with open(input_file, "r", encoding="utf-8") as fin:
        with open(output_file, "w", encoding="utf-8") as fout:
            for line in fin:
                if keyword in line:
                    fout.write(line)
                    count += 1

    print(f"Đã lọc {count} dòng chứa '{keyword}'")

# Test
filter_lines("data.txt", "filtered.txt", "Python")
```

Bài 5: CSV to Dictionary

```
def csv_to_dict(filename, key_field):
    """
    Chuyển CSV thành dictionary

    Args:
        filename (str): CSV file
        key_field (str): Field làm key

    Returns:
        dict: {key: {field: value}}
    """
    import csv

    result = {}

    with open(filename, "r", encoding="utf-8") as f:
        reader = csv.DictReader(f)
```

```
        for row in reader:
            key = row[key_field]
            result[key] = row

    return result

# Test với students.csv
students = csv_to_dict("students.csv", "name")
print(students["An"])  # {'name': 'An', 'age': '25', 'city': 'Hà Nội'}
```

Bài 6: Word frequency to file

```
def word_frequency_to_file(input_file, output_file):
    """
    Đếm tần suất từ và ghi ra file

    Format output: word,count
    """
    from collections import Counter

    # Đọc và đếm
    with open(input_file, "r", encoding="utf-8") as f:
        text = f.read().lower()
        words = text.split()
        freq = Counter(words)

    # Ghi ra file
    with open(output_file, "w", encoding="utf-8") as f:
        f.write("word,count\n")

        for word, count in freq.most_common():
            f.write(f"{word},{count}\n")

    print(f"Đã ghi {len(freq)} từ vào '{output_file}'")

# Test
word_frequency_to_file("input.txt", "frequency.csv")
```

Bài 7: Append log entries

```
def log_entry(log_file, message):
    """
    Thêm log entry vào file
    Format: [timestamp] message
    """
    from datetime import datetime
```

```
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    log_line = f"[{timestamp}] {message}\n"

    with open(log_file, "a", encoding="utf-8") as f:
        f.write(log_line)

# Test
log_entry("app.log", "Application started")
log_entry("app.log", "User logged in")
log_entry("app.log", "Data saved")
```

Bài 8: Convert CSV to JSON

```
import csv
import json

def csv_to_json(csv_file, json_file):
    """Chuyển CSV sang JSON"""
    with open(csv_file, "r", encoding="utf-8") as f:
        reader = csv.DictReader(f)
        data = list(reader)

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

    print(f"Đã convert '{csv_file}' → '{json_file}'")

# Test
csv_to_json("students.csv", "students.json")
```

---

10. Best Practices

10.1. ☑ DO

```
# ☑ Luôn dùng with statement
with open("file.txt", "r") as f:
    content = f.read()

# ☑ Luôn chỉ định encoding
with open("file.txt", "r", encoding="utf-8") as f:
    content = f.read()

# ☑ Kiểm tra file tồn tại
import os
if os.path.exists("file.txt"):
    with open("file.txt", "r") as f:
        content = f.read()
```

```
# ☑ Xử lý exceptions
try:
    with open("file.txt", "r") as f:
        content = f.read()
except FileNotFoundError:
    print("File không tồn tại!")

# ☑ Dùng csv module cho CSV files
import csv
with open("data.csv", "r") as f:
    reader = csv.DictReader(f)
    data = list(reader)

# ☑ Strip whitespace khi đọc
with open("file.txt", "r") as f:
    lines = [line.strip() for line in f]
```

10.2. ❌ DON'T

```
# ❌ Quên close()
f = open("file.txt", "r")
content = f.read()
# Quên f.close()

# ❌ Không chỉ định encoding
f = open("file.txt", "r")  # Dùng encoding mặc định

# ❌ Không xử lý errors
content = open("file.txt", "r").read()  # Lỗi nếu file không tồn tại

# ❌ Load toàn bộ file lớn vào memory
with open("huge_file.txt", "r") as f:
    content = f.read()  # Out of memory!

# ☑ Nên dùng iteration
with open("huge_file.txt", "r") as f:
    for line in f:  # Đọc từng dòng
        process(line)
```

---

11. Tổng Kết và Checklist

☑ Kiến thức cần nắm vững

File Operations:

- ☐ `open()` với modes: r, w, a, r+
- ☐ `.read()`, `.readline()`, `.readlines()`
- ☐ `.write()`, `.writelines()`

- ☐ Luôn dùng `with` statement

CSV Handling:

- ☐ Đọc CSV thủ công với `.split(',')`
- ☐ `csv.reader` và `csv.DictReader`
- ☐ `csv.writer` và `csv.DictWriter`
- ☐ Xử lý header

Error Handling:

- ☐ `FileNotFoundError`
- ☐ `PermissionError`
- ☐ `UnicodeDecodeError`

Assessment Applications:

- ☐ Bài 2: Đọc file strings
- ☐ Bài 3: Đọc CSV scores
- ☐ Bài 5: Đọc file codes

🎯 Lưu ý quan trọng

1. **LUÔN dùng `with`** - tự động close file
2. **LUÔN chỉ định encoding** - `encoding="utf-8"`
3. **`.strip()` khi đọc** - xóa whitespace thừa
4. **Kiểm tra file tồn tại** trước khi đọc
5. **Iterate từng dòng** với file lớn (đừng `.read()` hết)
6. **csv module** tốt hơn manual parsing
7. **Exception handling** cho production code

---

12. Câu Hỏi Thường Gặp (FAQ)

Q1: Tại sao phải dùng `encoding="utf-8"`?

A: Python 3 mặc định encoding khác nhau tùy OS. Chỉ định `utf-8` đảm bảo nhất quán.

```
# Windows: mặc định cp1252
# Linux/Mac: mặc định utf-8

# ❌ Không chỉ định - có thể lỗi với tiếng Việt
with open("file.txt", "r") as f:
    content = f.read()

# ☑ Chỉ định utf-8
with open("file.txt", "r", encoding="utf-8") as f:
    content = f.read()
```

Q2: `.read()` vs `.readlines()` vs iteration?

A:

- `.read()`: File nhỏ, cần toàn bộ text
- `.readlines()`: Cần random access
- **Iteration**: File lớn (BEST)

```
# File nhỏ
with open("small.txt", "r") as f:
    content = f.read()  # OK

# File lớn
with open("huge.txt", "r") as f:
    for line in f:  # ☑ Tiết kiệm memory
        process(line)
```

Q3: Mode 'w' vs 'a'?

A:

- `'w'`: **Xóa** nội dung cũ
- `'a'`: **Ghi tiếp** vào cuối

```
# 'w' - Xóa và ghi mới
with open("file.txt", "w") as f:
    f.write("New content")  # Xóa hết nội dung cũ

# 'a' - Ghi tiếp
with open("file.txt", "a") as f:
    f.write("Append this")  # Giữ nội dung cũ
```

Q4: Khi nào dùng csv module vs manual parsing?

A:

- **csv module**: CSV phức tạp (có quotes, commas trong data)
- **Manual**: CSV đơn giản

```
# CSV đơn giản
# name,age,city
# An,25,Hà Nội

# Manual OK
with open("data.csv", "r") as f:
    for line in f:
```

```
        parts = line.strip().split(',')

# CSV phức tạp
# name,description
# Product,"Contains, comma"

# ☑ Cần csv module
import csv
with open("data.csv", "r") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)  # Xử lý đúng commas trong quotes
```

Q5: Làm sao để đọc file từ dòng thứ N?

A: Dùng `itertools.islice()` hoặc skip với loop.

```
# Cách 1: Skip với loop
with open("file.txt", "r") as f:
    for i, line in enumerate(f):
        if i < 10:  # Skip 10 dòng đầu
            continue
        process(line)

# Cách 2: itertools.islice
from itertools import islice

with open("file.txt", "r") as f:
    for line in islice(f, 10, None):  # Từ dòng 10 trở đi
        process(line)
```
