# File Handling, Reading/Writing Files và CSV Basics

**Thời gian:** 3 tiếng (09:00-12:00)

**Mục tiêu:**

- Hiểu cách đọc và ghi files trong Python
- Làm việc với các định dạng file khác nhau
- Xử lý CSV files cơ bản
- Áp dụng file handling vào các dự án thực tế

**Kiến thức cần có:**

- Functions và error handling
- Lists, dictionaries
- String manipulation
- Lambda functions

## 1. File Handling Cơ bản

### 1.1 Hiểu về Files và Paths

In [ ]:
```python
# Hiểu về file paths và current directory
import os

print("=== THÔNG TIN THƯ MỤC HIỆN TẠI ===")
print(f"Thư mục hiện tại: {os.getcwd()}")
print(f"Danh sách files: {os.listdir('.')}")

# Kiểm tra file có tồn tại không
def kiem_tra_file(ten_file):
    """Kiểm tra file có tồn tại không"""
    if os.path.exists(ten_file):
        print(f"✅ File '{ten_file}' tồn tại")
        print(f"   Kích thước: {os.path.getsize(ten_file)} bytes")
    else:
        print(f"❌ File '{ten_file}' không tồn tại")

# Test
kiem_tra_file("test.txt")
kiem_tra_file("data.csv")
```

### 1.2 Mở và đóng files

In [ ]:
```python
# Các modes mở file
print("=== CÁC MODES MỞ FILE ===")
print("'r'  - Read (đọc) - mặc định")
print("'w'  - Write (ghi) - xóa nội dung cũ")
print("'a'  - Append (thêm) - giữ nội dung cũ")
print("'r+' - Read + Write")
print("'x'  - Create - tạo file mới (lỗi nếu đã tồn tại)")

# Cách MỞ và ĐÓNG file cơ bản (không khuyến khích)
def mo_file_co_ban(ten_file):
    """Cách mở file cơ bản - CẦN PHẢI đóng file"""
    try:
        file = open(ten_file, 'r', encoding='utf-8')
        noi_dung = file.read()
        file.close()  # QUAN TRỌNG: phải đóng file
        return noi_dung
    except FileNotFoundError:
        print(f"File {ten_file} không tồn tại")
        return None

# Cách KHUYẾN KHÍCH: sử dụng context manager (with)
def mo_file_an_toan(ten_file):
    """Cách mở file an toàn - tự động đóng file"""
    try:
        with open(ten_file, 'r', encoding='utf-8') as file:
            noi_dung = file.read()
            return noi_dung
        # File tự động đóng khi ra khỏi block with
    except FileNotFoundError:
        print(f"File {ten_file} không tồn tại")
        return None
    except Exception as e:
        print(f"Lỗi khi đọc file: {e}")
        return None

print("\n=== LUÔN SỬ DỤNG 'with' STATEMENT ===")
print("Tự động đóng file, an toàn hơn!")
```

# 2. Writing Files (Ghi Files)

## 2.1 Ghi file text cơ bản

In [ ]:
```python
# Ghi file text đơn giản
def tao_file_text(ten_file, noi_dung):
    """Tạo file text với nội dung cho trước"""
    try:
        with open(ten_file, 'w', encoding='utf-8') as file:
            file.write(noi_dung)
        print(f"✅ Đã tạo file '{ten_file}' thành công")
    except Exception as e:
        print(f"❌ Lỗi khi tạo file: {e}")

# Thêm nội dung vào file (append)
```

```
def them_noi_dung(ten_file, noi_dung_them):
    """Thêm nội dung vào cuối file"""
    try:
        with open(ten_file, 'a', encoding='utf-8') as file:
            file.write(noi_dung_them)
        print(f"✅ Đã thêm nội dung vào '{ten_file}'")
    except Exception as e:
        print(f"❌ Lỗi khi thêm nội dung: {e}")

# Test writing files
noi_dung_mau = """Xin chào! Đây là file text đầu tiên.
Tôi đang học Python file handling.
Rất thú vị và hữu ích!
"""

tao_file_text("hello.txt", noi_dung_mau)
them_noi_dung("hello.txt", "\nDòng này được thêm sau!\n")

# Kiểm tra kết quả
kiem_tra_file("hello.txt")
```

## 2.2 Ghi nhiều dòng và lists

In [ ]:

```
# Ghi list thành file
def ghi_list_ra_file(ten_file, danh_sach, title="Danh sách"):
    """Ghi list ra file với format đẹp"""
    try:
        with open(ten_file, 'w', encoding='utf-8') as file:
            file.write(f"=== {title.upper()} ===\n\n")
            for i, item in enumerate(danh_sach, 1):
                file.write(f"{i}. {item}\n")
            file.write(f"\nTổng cộng: {len(danh_sach)} items")
        print(f"✅ Đã ghi {len(danh_sach)} items vào '{ten_file}'")
    except Exception as e:
        print(f"❌ Lỗi: {e}")

# Ghi dictionary ra file
def ghi_dict_ra_file(ten_file, tu_dien, title="Từ điển"):
    """Ghi dictionary ra file với format key: value"""
    try:
        with open(ten_file, 'w', encoding='utf-8') as file:
            file.write(f"=== {title.upper()} ===\n\n")
            for key, value in tu_dien.items():
                file.write(f"{key}: {value}\n")
            file.write(f"\nTổng cộng: {len(tu_dien)} items")
        print(f"✅ Đã ghi dictionary vào '{ten_file}'")
    except Exception as e:
        print(f"❌ Lỗi: {e}")

# Test với data mẫu
danh_sach_mua_sam = [
    "Gạo", "Thịt heo", "Rau cải", "Trứng gà",
    "Sữa tươi", "Bánh mì", "Cà chua", "Hành tây"
]
```

```
thong_tin_sinh_vien = {
    "Họ tên": "Nguyễn Văn A",
    "Tuổi": 20,
    "Lớp": "Python101",
    "Điểm TB": 8.5,
    "Quê quán": "Hà Nội"
}

ghi_list_ra_file("shopping_list.txt", danh_sach_mua_sam, "Danh sách mua sắm")
ghi_dict_ra_file("student_info.txt", thong_tin_sinh_vien, "Thông tin sinh viên")
```

## 🔥 TODO 1: Thực hành Writing Files

**Yêu cầu:**

1. Viết function `luu_lich_su_tinh_toan(ten_file, danh_sach_phep_tinh)`
2. Viết function `luu_diem_sinh_vien(ten_file, dict_sinh_vien)`
3. Viết function `tao_bao_cao_hang_ngay(ten_file, hoat_dong_list)`
4. Test với dữ liệu mẫu

In [ ]:

```
# TODO 1: Viết code ở đây
from datetime import datetime

def luu_lich_su_tinh_toan(ten_file, danh_sach_phep_tinh):
    """Lưu lịch sử tính toán với timestamp"""
    try:
        with open(ten_file, 'w', encoding='utf-8') as file:
            file.write("=== LỊCH SỬ TÍNH TOÁN ===\n")
            file.write(f"Ngày: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n\n")
            for i, phep_tinh in enumerate(danh_sach_phep_tinh, 1):
                file.write(f"{i}. {phep_tinh}\n")
            file.write(f"\nTổng cộng: {len(danh_sach_phep_tinh)} phép tính")
        print(f"✅ Đã lưu {len(danh_sach_phep_tinh)} phép tính vào {ten_file}")
    except Exception as e:
        print(f"❌ Lỗi: {e}")

def luu_diem_sinh_vien(ten_file, dict_sinh_vien):
    """Lưu điểm sinh viên theo format đẹp"""
    try:
        with open(ten_file, 'w', encoding='utf-8') as file:
            file.write("=== BẢNG ĐIỂM SINH VIÊN ===\n")
            file.write(f"Ngày tạo: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n

            for ho_ten, diem_dict in dict_sinh_vien.items():
                diem_tb = sum(diem_dict.values()) / len(diem_dict)

                if diem_tb >= 8.5:
                    xep_loai = "Giỏi"
                elif diem_tb >= 7:
                    xep_loai = "Khá"
                elif diem_tb >= 5:
                    xep_loai = "Trung bình"
                else:
```

```
                    xep_loai = "Yếu"

                file.write(f"Sinh viên: {ho_ten}\n")
                for mon, diem in diem_dict.items():
                    file.write(f"  {mon.capitalize()}: {diem}\n")
                file.write(f"  Điểm TB: {diem_tb:.2f}\n")
                file.write(f"  Xếp loại: {xep_loai}\n")
                file.write("-" * 30 + "\n")

        print(f"✅ Đã lưu điểm {len(dict_sinh_vien)} sinh viên vào {ten_file}")
    except Exception as e:
        print(f"❌ Lỗi: {e}")

def tao_bao_cao_hang_ngay(ten_file, hoat_dong_list):
    """Tạo báo cáo hoạt động hàng ngày"""
    try:
        with open(ten_file, 'w', encoding='utf-8') as file:
            file.write("=== BÁO CÁO HOẠT ĐỘNG HÀNG NGÀY ===\n")
            file.write(f"Ngày: {datetime.now().strftime('%Y-%m-%d')}\n")
            file.write(f"Thời gian tạo báo cáo: {datetime.now().strftime('%H:%M:%S'

            file.write("DANH SÁCH HOẠT ĐỘNG:\n")
            for i, hoat_dong in enumerate(hoat_dong_list, 1):
                file.write(f"{i}. {hoat_dong}\n")

            file.write(f"\nTổng số hoạt động: {len(hoat_dong_list)}\n")
            file.write(f"Thời gian trung bình/hoạt động: {480/len(hoat_dong_list):.

        print(f"✅ Đã tạo báo cáo với {len(hoat_dong_list)} hoạt động")
    except Exception as e:
        print(f"❌ Lỗi: {e}")

# Test data
lich_su_calculator = [
    "5 + 3 = 8",
    "10 * 2 = 20",
    "√16 = 4",
    "2^8 = 256"
]

diem_lop_python = {
    "Nguyễn Văn A": {"toan": 8, "ly": 7, "hoa": 9},
    "Trần Thị B": {"toan": 9, "ly": 8, "hoa": 8},
    "Lê Văn C": {"toan": 6, "ly": 7, "hoa": 6}
}

hoat_dong_hom_nay = [
    "09:00 - Học Python file handling",
    "10:30 - Thực hành đọc/ghi files",
    "11:00 - Làm bài tập TODO",
    "11:30 - Review code với bạn"
]

# Test functions
luu_lich_su_tinh_toan("calculator_history.txt", lich_su_calculator)
```

```
luu_diem_sinh_vien("student_grades.txt", diem_lop_python)
tao_bao_cao_hang_ngay("daily_report.txt", hoat_dong_hom_nay)
```

# 3. Reading Files (Đọc Files)

## 3.1 Các cách đọc files

In [ ]:
```
# Các phương thức đọc file
def demo_cac_cach_doc_file(ten_file):
    """Demo các cách đọc file khác nhau"""
    try:
        print(f"=== ĐỌC FILE: {ten_file} ===")

        # Cách 1: Đọc toàn bộ file
        with open(ten_file, 'r', encoding='utf-8') as file:
            toan_bo = file.read()
            print(f"1. Đọc toàn bộ ({len(toan_bo)} ký tự):")
            print(f"   {toan_bo[:100]}..." if len(toan_bo) > 100 else f"   {toan_bo

        # Cách 2: Đọc từng dòng thành list
        with open(ten_file, 'r', encoding='utf-8') as file:
            cac_dong = file.readlines()
            print(f"\n2. Đọc thành list ({len(cac_dong)} dòng):")
            for i, dong in enumerate(cac_dong[:3], 1):
                print(f"   Dòng {i}: {dong.strip()}")

        # Cách 3: Đọc từng dòng với vòng lặp
        print(f"\n3. Đọc từng dòng:")
        with open(ten_file, 'r', encoding='utf-8') as file:
            for i, dong in enumerate(file, 1):
                print(f"   {i}: {dong.strip()}")
                if i >= 3:  # Chỉ hiển thị 3 dòng đầu
                    print("   ...")
                    break

    except FileNotFoundError:
        print(f"❌ File {ten_file} không tồn tại")
    except Exception as e:
        print(f"❌ Lỗi: {e}")

# Test với file vừa tạo
demo_cac_cach_doc_file("hello.txt")
```

## 3.2 Xử lý và phân tích nội dung file

In [ ]:
```
# Đọc và phân tích file text
def phan_tich_file_text(ten_file):
    """Phân tích thống kê file text"""
    try:
        with open(ten_file, 'r', encoding='utf-8') as file:
            noi_dung = file.read()
```

```
        # Thống kê cơ bản
        so_ky_tu = len(noi_dung)
        so_dong = noi_dung.count('\n') + 1
        so_tu = len(noi_dung.split())
        so_doan_van = noi_dung.count('\n\n') + 1

        print(f"=== PHÂN TÍCH FILE: {ten_file} ===")
        print(f"📝 Số ký tự: {so_ky_tu:,}")
        print(f"📄 Số dòng: {so_dong:,}")
        print(f"🔤 Số từ: {so_tu:,}")
        print(f"📑 Số đoạn văn: {so_doan_van}")

        # Từ xuất hiện nhiều nhất
        tu_list = noi_dung.lower().split()
        tu_dem = {}
        for tu in tu_list:
            tu = tu.strip('.,!?"()[]')
            tu_dem[tu] = tu_dem.get(tu, 0) + 1

        top_tu = sorted(tu_dem.items(), key=lambda x: x[1], reverse=True)[:5]
        print(f"\n🔥 Top 5 từ phổ biến:")
        for tu, so_lan in top_tu:
            print(f"   '{tu}': {so_lan} lần")

    except Exception as e:
        print(f"❌ Lỗi: {e}")

# Đọc file có cấu trúc (key: value)
def doc_file_cau_truc(ten_file):
    """Đọc file có cấu trúc key: value thành dictionary"""
    du_lieu = {}
    try:
        with open(ten_file, 'r', encoding='utf-8') as file:
            for dong in file:
                dong = dong.strip()
                if ':' in dong and not dong.startswith('==='):
                    key, value = dong.split(':', 1)
                    du_lieu[key.strip()] = value.strip()

        print(f"\n=== DỮ LIỆU TỪ {ten_file} ===")
        for key, value in du_lieu.items():
            print(f"{key}: {value}")

        return du_lieu
    except Exception as e:
        print(f"❌ Lỗi: {e}")
        return {}

# Test
phan_tich_file_text("hello.txt")
student_data = doc_file_cau_truc("student_info.txt")
```

## 🔥 TODO 2: Thực hành Reading Files

**Yêu cầu:**

1. Viết function `doc_va_loc_log_file(ten_file, keyword)` - tìm dòng chứa keyword
2. Viết function `dem_tu_trong_file(ten_file)` - đếm tần suất từ
3. Viết function `tim_dong_dai_nhat(ten_file)` - tìm dòng dài nhất
4. Viết function `doc_config_file(ten_file)` - đọc file config

In [ ]:
```python
# TODO 2: Viết code ở đây

def doc_va_loc_log_file(ten_file, keyword):
    """Đọc file log và lọc các dòng chứa keyword"""
    try:
        ket_qua = []
        with open(ten_file, 'r', encoding='utf-8') as file:
            for so_dong, dong in enumerate(file, 1):
                if keyword.lower() in dong.lower():
                    ket_qua.append((so_dong, dong.strip()))

        print(f"Tìm thấy {len(ket_qua)} dòng chứa '{keyword}':")
        for so_dong, noi_dung in ket_qua:
            print(f"  Dòng {so_dong}: {noi_dung}")

        return ket_qua
    except Exception as e:
        print(f"❌ Lỗi: {e}")
        return []

def dem_tu_trong_file(ten_file):
    """Đếm tần suất xuất hiện của mỗi từ"""
    try:
        with open(ten_file, 'r', encoding='utf-8') as file:
            noi_dung = file.read().lower()

        # Loại bỏ dấu câu và tách từ
        import re
        tu_list = re.findall(r'\b\w+\b', noi_dung)

        # Đếm tần suất
        tu_dem = {}
        for tu in tu_list:
            tu_dem[tu] = tu_dem.get(tu, 0) + 1

        # Sắp xếp theo tần suất giảm dần
        tu_sap_xep = sorted(tu_dem.items(), key=lambda x: x[1], reverse=True)

        print(f"Tìm thấy {len(tu_sap_xep)} từ khác nhau:")
        for tu, so_lan in tu_sap_xep[:10]:  # Hiển thị top 10
            print(f"  '{tu}': {so_lan} lần")

        return dict(tu_sap_xep)
    except Exception as e:
        print(f"❌ Lỗi: {e}")
        return {}

def tim_dong_dai_nhat(ten_file):
```

```
    """Tìm dòng dài nhất trong file"""
    try:
        dong_dai_nhat = ""
        so_dong_dai_nhat = 0
        do_dai_max = 0

        with open(ten_file, 'r', encoding='utf-8') as file:
            for so_dong, dong in enumerate(file, 1):
                dong = dong.strip()
                if len(dong) > do_dai_max:
                    do_dai_max = len(dong)
                    dong_dai_nhat = dong
                    so_dong_dai_nhat = so_dong

        print(f"Dòng dài nhất (dòng {so_dong_dai_nhat}, {do_dai_max} ký tự):")
        print(f"  '{dong_dai_nhat}'")

        return (so_dong_dai_nhat, dong_dai_nhat, do_dai_max)
    except Exception as e:
        print(f"❌ Lỗi: {e}")
        return (0, "", 0)

def doc_config_file(ten_file):
    """Đọc file config dạng key=value"""
    try:
        config = {}
        with open(ten_file, 'r', encoding='utf-8') as file:
            for so_dong, dong in enumerate(file, 1):
                dong = dong.strip()
                # Bỏ qua dòng comment và dòng trống
                if dong and not dong.startswith('#'):
                    if '=' in dong:
                        key, value = dong.split('=', 1)
                        config[key.strip()] = value.strip()
                    else:
                        print(f"⚠️ Dòng {so_dong} không đúng format: {dong}")

        print(f"Đọc được {len(config)} cài đặt:")
        for key, value in config.items():
            print(f"  {key} = {value}")

        return config
    except Exception as e:
        print(f"❌ Lỗi: {e}")
        return {}

# Tạo test files
log_content = """2024-01-15 09:00:01 INFO User login successful
2024-01-15 09:01:23 WARNING Database connection slow
2024-01-15 09:02:45 ERROR Failed to save data
2024-01-15 09:03:12 INFO User logout
2024-01-15 09:04:33 DEBUG Processing request
2024-01-15 09:05:44 ERROR Connection timeout
"""

config_content = """# Calculator Settings
```

```
decimal_places=2
auto_save=true
theme=dark
language=vi
# Advanced settings
scientific_mode=false
"""

# Tạo test files
with open("app.log", "w", encoding="utf-8") as f:
    f.write(log_content)

with open("config.ini", "w", encoding="utf-8") as f:
    f.write(config_content)

# Test functions
print("=== TEST READING FUNCTIONS ===")
error_lines = doc_va_loc_log_file("app.log", "ERROR")
print(f"\nTìm thấy {len(error_lines)} dòng ERROR")

word_count = dem_tu_trong_file("hello.txt")
print(f"\nFile có {len(word_count)} từ khác nhau")

longest = tim_dong_dai_nhat("app.log")
print(f"\nDòng dài nhất: dòng {longest[0]}")

config = doc_config_file("config.ini")
print(f"\nConfig loaded: {len(config)} settings")
```

# 4. CSV Basics - Làm việc với CSV Files

## 4.1 Hiểu về CSV Format

In [ ]:

```
# CSV (Comma-Separated Values) là format rất phổ biến
print("=== CSV FORMAT BASICS ===")
print("CSV = Comma-Separated Values")
print("Mỗi dòng = 1 record")
print("Các cột cách nhau bởi dấu phẩy (,)")
print("Dòng đầu thường là header (tên cột)")

# Tạo CSV file đơn giản
def tao_csv_don_gian(ten_file, data, headers):
    """Tạo CSV file từ data và headers"""
    try:
        with open(ten_file, 'w', encoding='utf-8') as file:
            # Ghi header
            file.write(','.join(headers) + '\n')

            # Ghi data
            for row in data:
                file.write(','.join(map(str, row)) + '\n')

        print(f"✅ Đã tạo CSV file: {ten_file}")
    except Exception as e:
```

```
        print(f"❌ Lỗi: {e}")

# Data mẫu
sinh_vien_headers = ['STT', 'Ho_Ten', 'Tuoi', 'Lop', 'Diem_TB']
sinh_vien_data = [
    [1, 'Nguyen Van A', 20, 'Python101', 8.5],
    [2, 'Tran Thi B', 19, 'Python101', 9.0],
    [3, 'Le Van C', 21, 'Python102', 7.5],
    [4, 'Pham Thi D', 20, 'Python101', 8.8],
    [5, 'Hoang Van E', 22, 'Python102', 6.9]
]

tao_csv_don_gian('students.csv', sinh_vien_data, sinh_vien_headers)

# Xem kết quả
print("\n=== NỘI DUNG CSV FILE ===")
with open('students.csv', 'r', encoding='utf-8') as file:
    for i, line in enumerate(file, 1):
        print(f"Dòng {i}: {line.strip()}")
```

## 4.2 Đọc CSV Files

In [ ]:

```
# Đọc CSV thủ công (cách cơ bản)
def doc_csv_thu_cong(ten_file):
    """Đọc CSV file bằng cách thủ công"""
    try:
        with open(ten_file, 'r', encoding='utf-8') as file:
            lines = file.readlines()

        # Lấy header
        headers = lines[0].strip().split(',')
        print(f"Headers: {headers}")

        # Lấy data
        data = []
        for line in lines[1:]:
            row = line.strip().split(',')
            data.append(row)

        print(f"\nData ({len(data)} rows):")
        for i, row in enumerate(data[:3], 1):  # Chỉ hiển thị 3 dòng đầu
            print(f"  Row {i}: {row}")

        return headers, data

    except Exception as e:
        print(f"❌ Lỗi: {e}")
        return [], []

# Đọc CSV thành list of dictionaries
def doc_csv_thanh_dict(ten_file):
    """Đọc CSV thành list of dictionaries (dễ sử dụng hơn)"""
    try:
        with open(ten_file, 'r', encoding='utf-8') as file:
            lines = file.readlines()
```

```
        headers = lines[0].strip().split(',')
        data_list = []

        for line in lines[1:]:
            values = line.strip().split(',')
            row_dict = {}
            for i, header in enumerate(headers):
                row_dict[header] = values[i] if i < len(values) else ''
            data_list.append(row_dict)

        return data_list

    except Exception as e:
        print(f"❌ Lỗi: {e}")
        return []

# Test đọc CSV
print("=== ĐỌC CSV THỦ CÔNG ===")
headers, data = doc_csv_thu_cong('students.csv')

print("\n=== ĐỌC CSV THÀNH DICTIONARY ===")
students_dict = doc_csv_thanh_dict('students.csv')
for i, student in enumerate(students_dict[:2], 1):
    print(f"Student {i}: {student}")
```

## 4.3 Sử dụng csv module (Khuyến khích)

In [ ]:

```
# Sử dụng csv module (chính thức và mạnh mẽ hơn)
import csv

def ghi_csv_voi_module(ten_file, data, headers):
    """Ghi CSV sử dụng csv module"""
    try:
        with open(ten_file, 'w', newline='', encoding='utf-8') as file:
            writer = csv.writer(file)
            writer.writerow(headers)  # Ghi header
            writer.writerows(data)    # Ghi tất cả data
        print(f"✅ Đã ghi CSV với module: {ten_file}")
    except Exception as e:
        print(f"❌ Lỗi: {e}")

def doc_csv_voi_module(ten_file):
    """Đọc CSV sử dụng csv module"""
    try:
        with open(ten_file, 'r', encoding='utf-8') as file:
            reader = csv.reader(file)
            headers = next(reader)  # Đọc dòng đầu (headers)
            data = list(reader)     # Đọc tất cả dòng còn lại

        print(f"Headers: {headers}")
        print(f"Data: {len(data)} rows")
        return headers, data
    except Exception as e:
        print(f"❌ Lỗi: {e}")
```

```
        return [], []

def doc_csv_dict_reader(ten_file):
    """Đọc CSV sử dụng DictReader (tự động tạo dictionary)"""
    try:
        with open(ten_file, 'r', encoding='utf-8') as file:
            reader = csv.DictReader(file)
            data_list = list(reader)

        print(f"Đọc được {len(data_list)} records")
        return data_list
    except Exception as e:
        print(f"❌ Lỗi: {e}")
        return []

# Test csv module
print("=== SỬ DỤNG CSV MODULE ===")

# Ghi CSV với module
ban_hang_headers = ['San_pham', 'So_luong', 'Gia', 'Thanh_tien']
ban_hang_data = [
    ['Laptop Dell', 2, 15000000, 30000000],
    ['Mouse Logitech', 5, 500000, 2500000],
    ['Keyboard Mechanical', 3, 2000000, 6000000],
    ['Monitor 24inch', 1, 5000000, 5000000]
]

ghi_csv_voi_module('sales.csv', ban_hang_data, ban_hang_headers)

# Đọc CSV với module
headers, data = doc_csv_voi_module('sales.csv')

# Đọc CSV với DictReader
print("\n=== SỬ DỤNG DICTREADER ===")
sales_data = doc_csv_dict_reader('sales.csv')
for item in sales_data[:2]:
    print(f"  {item}")
```

## 🔥 TODO 3: Thực hành CSV Operations

**Yêu cầu:**

1. Viết function `phan_tich_diem_csv(ten_file)` - phân tích điểm từ CSV
2. Viết function `loc_sinh_vien_gioi(ten_file_input, ten_file_output)`
3. Viết function `tinh_thong_ke_ban_hang(ten_file)`
4. Viết function `them_cot_xep_loai(ten_file_input, ten_file_output)`

In [ ]:
```
# TODO 3: Viết code ở đây
import csv

def phan_tich_diem_csv(ten_file):
    """Phân tích điểm sinh viên từ CSV file"""
    # Viết code để:
```

```
    # - Đọc CSV file với DictReader
    # - Tính điểm trung bình của lớp
    # - Tìm sinh viên có điểm cao nhất/thấp nhất
    # - Đếm số sinh viên theo từng mức điểm (Giỏi, Khá, TB, Yếu)
    # - In báo cáo thống kê
    pass

def loc_sinh_vien_gioi(ten_file_input, ten_file_output):
    """Lọc sinh viên giỏi (điểm >= 8.0) và ghi ra file mới"""
    # Viết code để:
    # - Đọc CSV input
    # - Lọc sinh viên có điểm >= 8.0
    # - Ghi ra CSV output với cùng format
    pass

def tinh_thong_ke_ban_hang(ten_file):
    """Tính thống kê bán hàng từ CSV"""
    # Viết code để:
    # - Đọc dữ liệu bán hàng
    # - Tính tổng doanh thu
    # - Tìm sản phẩm bán chạy nhất
    # - Tính trung bình doanh thu per sản phẩm
    pass

def them_cot_xep_loai(ten_file_input, ten_file_output):
    """Thêm cột xếp loại vào CSV sinh viên"""
    # Viết code để:
    # - Đọc CSV sinh viên
    # - Thêm cột 'Xep_loai' based on điểm TB
    # - Ghi ra file mới với cột bổ sung
    # Quy tắc: >= 8.5: Giỏi, >= 7: Khá, >= 5: TB, < 5: Yếu
    pass

# Tạo thêm data để test
more_students = [
    [6, 'Nguyen Thi F', 19, 'Python103', 9.2],
    [7, 'Tran Van G', 20, 'Python103', 4.5],
    [8, 'Le Thi H', 21, 'Python101', 7.8],
    [9, 'Pham Van I', 22, 'Python102', 8.9],
    [10, 'Hoang Thi J', 20, 'Python103', 6.2]
]

# Thêm vào CSV
with open('students.csv', 'a', encoding='utf-8') as file:
    for student in more_students:
        file.write(','.join(map(str, student)) + '\n')

print("=== TEST CSV ANALYSIS FUNCTIONS ===")
phan_tich_diem_csv('students.csv')
loc_sinh_vien_gioi('students.csv', 'students_gioi.csv')
tinh_thong_ke_ban_hang('sales.csv')
them_cot_xep_loai('students.csv', 'students_xep_loai.csv')
```

# 5. Practical Applications - Ứng dụng thực tế

## 5.1 File-based Data Management System

In [ ]:
```python
# Hệ thống quản lý dữ liệu dựa trên files
import csv
import os
from datetime import datetime

class FileDataManager:
    """Class quản lý dữ liệu sử dụng files"""

    def __init__(self, data_folder="data"):
        self.data_folder = data_folder
        self.tao_thu_muc_neu_chua_co()

    def tao_thu_muc_neu_chua_co(self):
        """Tạo thư mục data nếu chưa có"""
        if not os.path.exists(self.data_folder):
            os.makedirs(self.data_folder)
            print(f"✅ Đã tạo thư mục: {self.data_folder}")

    def get_file_path(self, ten_file):
        """Lấy đường dẫn đầy đủ của file"""
        return os.path.join(self.data_folder, ten_file)

    def luu_dict_ra_json_file(self, ten_file, data_dict):
        """Lưu dictionary ra file dạng JSON đơn giản"""
        file_path = self.get_file_path(ten_file)
        try:
            with open(file_path, 'w', encoding='utf-8') as file:
                # Ghi dictionary theo format đơn giản
                file.write(f"# Saved at: {datetime.now()}\n")
                for key, value in data_dict.items():
                    file.write(f"{key}={value}\n")
            print(f"✅ Đã lưu {len(data_dict)} items vào {ten_file}")
        except Exception as e:
            print(f"❌ Lỗi khi lưu: {e}")

    def doc_dict_tu_file(self, ten_file):
        """Đọc dictionary từ file"""
        file_path = self.get_file_path(ten_file)
        data_dict = {}
        try:
            with open(file_path, 'r', encoding='utf-8') as file:
                for line in file:
                    line = line.strip()
                    if '=' in line and not line.startswith('#'):
                        key, value = line.split('=', 1)
                        data_dict[key] = value
            print(f"✅ Đã đọc {len(data_dict)} items từ {ten_file}")
            return data_dict
        except FileNotFoundError:
            print(f"❌ File {ten_file} không tồn tại")
            return {}
        except Exception as e:
            print(f"❌ Lỗi: {e}")
```

```
            return {}

    def backup_file(self, ten_file):
        """Tạo backup của file"""
        file_path = self.get_file_path(ten_file)
        if os.path.exists(file_path):
            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
            backup_name = f"{ten_file}.backup_{timestamp}"
            backup_path = self.get_file_path(backup_name)

            try:
                with open(file_path, 'r', encoding='utf-8') as original:
                    with open(backup_path, 'w', encoding='utf-8') as backup:
                        backup.write(original.read())
                print(f"✅ Đã backup: {backup_name}")
                return backup_name
            except Exception as e:
                print(f"❌ Lỗi backup: {e}")
                return None

    def list_files(self, mo_rong=""):
        """Liệt kê các files trong thư mục data"""
        try:
            files = os.listdir(self.data_folder)
            if mo_rong:
                files = [f for f in files if f.endswith(mo_rong)]

            print(f"\n=== FILES TRONG {self.data_folder} ===")
            for file in sorted(files):
                file_path = self.get_file_path(file)
                size = os.path.getsize(file_path)
                print(f"📄 {file} ({size} bytes)")

            return files
        except Exception as e:
            print(f"❌ Lỗi: {e}")
            return []

# Test File Data Manager
print("=== TEST FILE DATA MANAGER ===")
dm = FileDataManager()

# Test lưu và đọc data
app_settings = {
    'theme': 'dark',
    'language': 'vi',
    'auto_save': 'true',
    'decimal_places': '2'
}

dm.luu_dict_ra_json_file('app_settings.conf', app_settings)
loaded_settings = dm.doc_dict_tu_file('app_settings.conf')
print(f"Loaded settings: {loaded_settings}")

# Test backup
backup_name = dm.backup_file('app_settings.conf')
```

```
# List files
dm.list_files()
```

## 🔥 TODO 4: Xây dựng Student Management System

**Yêu cầu:**

1. Sử dụng FileDataManager để lưu trữ dữ liệu
2. Tạo class `StudentManager` với các chức năng:
   - Thêm/sửa/xóa sinh viên
   - Export/Import CSV
   - Tìm kiếm sinh viên
   - Thống kê điểm số
3. Menu-driven interface
4. Auto-backup mỗi khi có thay đổi

In [ ]:
```
# TODO 4: Student Management System

class StudentManager:
    """Hệ thống quản lý sinh viên sử dụng files"""

    def __init__(self):
        self.data_manager = FileDataManager("student_data")
        self.students_file = "students.csv"
        self.settings_file = "settings.conf"
        self.load_settings()

    def load_settings(self):
        """Load cài đặt hệ thống"""
        # Viết code để load settings
        pass

    def them_sinh_vien(self, ho_ten, tuoi, lop, diem_toan, diem_ly, diem_hoa):
        """Thêm sinh viên mới"""
        # Viết code để:
        # - Tính điểm TB
        # - Thêm vào CSV file
        # - Auto backup nếu cần
        pass

    def tim_sinh_vien(self, keyword):
        """Tìm sinh viên theo tên hoặc lớp"""
        # Viết code để tìm kiếm trong CSV
        pass

    def cap_nhat_diem(self, ho_ten, mon_hoc, diem_moi):
        """Cập nhật điểm của sinh viên"""
        # Viết code để update CSV file
        pass

    def xoa_sinh_vien(self, ho_ten):
```

```
        """Xóa sinh viên"""
        # Viết code để xóa khỏi CSV
        pass

    def thong_ke_lop(self, ten_lop=None):
        """Thống kê điểm theo lớp"""
        # Viết code thống kê
        pass

    def export_excel_format(self, ten_file_output):
        """Export sang format Excel-friendly CSV"""
        # Viết code export với encoding phù hợp
        pass

    def import_tu_csv(self, ten_file_input):
        """Import sinh viên từ CSV file khác"""
        # Viết code import và merge data
        pass

    def menu_chinh(self):
        """Menu chính của hệ thống"""
        while True:
            print("\n" + "="*50)
            print("      STUDENT MANAGEMENT SYSTEM")
            print("="*50)
            print("1. Thêm sinh viên")
            print("2. Tìm sinh viên")
            print("3. Cập nhật điểm")
            print("4. Xóa sinh viên")
            print("5. Thống kê lớp")
            print("6. Export CSV")
            print("7. Import CSV")
            print("8. Backup dữ liệu")
            print("9. Exit")
            print("-"*50)

            choice = input("Chọn chức năng (1-9): ").strip()

            if choice == '1':
                print("\n=== THÊM SÁCH MỚI ===")
                ten_sach = input("Tên sách: ")
                tac_gia = input("Tác giả: ")
                nam_xb = input("Năm xuất bản: ")
                the_loai = input("Thể loại: ")
                so_luong = input("Số lượng: ")
                self.them_sach(ten_sach, tac_gia, nam_xb, the_loai, so_luong)

            elif choice == '2':
                keyword = input("\nNhập từ khóa tìm kiếm: ")
                self.tim_sach(keyword)

            elif choice == '3':
                self.thong_ke_the_loai()

            elif choice == '4':
                self.export_bao_cao()
```

```
            elif choice == '5':
                self.backup_du_lieu()

            elif choice == '6':
                self.export_excel_format()

            elif choice == '7':
                self.import_tu_csv()

            elif choice == '8':
                self.backup_du_lieu()

            elif choice == '9':
                print("Cảm ơn bạn đã sử dụng hệ thống!")
                self.ghi_log("Thoát hệ thống")
                break

            else:
                print("Lựa chọn không hợp lệ!")

# Demo hệ thống
print("=== DEMO LIBRARY MANAGEMENT SYSTEM ===")
library = LibraryManager()

# Test một số chức năng
library.tim_sach("Python")
library.thong_ke_the_loai()
library.export_bao_cao()

print("\n📚 Hệ thống thư viện đã sẵn sàng!")
print("Chạy library.menu_chinh() để sử dụng menu tương tác.")
```