Bài tập thực hành: Student Management System

Tuần 27 - Thứ 3 (14:00-17:00)

Mục tiêu

- Áp dụng List Methods (append, remove, pop, insert)
- Sử dụng List Comprehensions để xử lý dữ liệu
- Thao tác với Nested Lists
- Sorting và Filtering dữ liệu học sinh

Tình huống

Bạn là quản lý của một lớp học và cần xây dựng hệ thống quản lý học sinh đơn giản để:

- Lưu trữ thông tin học sinh
- Thêm/xóa học sinh
- Tính điểm trung bình
- Sắp xếp và lọc dữ liệu

Phần 1: Khởi tạo dữ liệu học sinh

Chúng ta sẽ lưu thông tin học sinh dưới dạng nested list: `[tên, tuổi, [điểm_toán, điểm_lý, điểm_hóa]]`

In [ ]:
```
# Dữ liệu ban đầu - 5 học sinh
students = [
    ['Nguyễn Văn An', 16, [8.5, 7.0, 9.0]],
    ['Trần Thị Bình', 17, [9.0, 8.5, 7.5]],
    ['Lê Văn Chi', 16, [7.0, 8.0, 8.5]],
    ['Phạm Thị Dung', 17, [9.5, 9.0, 8.0]],
    ['Hoàng Văn Em', 16, [6.5, 7.5, 7.0]]
]

print("=== DANH SÁCH HỌC SINH BAN ĐẦU ===")
for i, student in enumerate(students):
    name, age, grades = student
    print(f"{i+1}. {name} ({age} tuổi) - Điểm: {grades}")

print(f"\nTổng số học sinh: {len(students)}")
```

Phần 2: Thêm học sinh mới

Bài tập 1: Sử dụng `append()` để thêm học sinh mới

In [ ]:
```python
# TODO 1: Thêm 2 học sinh mới vào danh sách
# Học sinh 1: 'Võ Thị Giang', 17 tuổi, điểm [8.0, 8.5, 9.0]
# Học sinh 2: 'Đặng Văn Hùng', 16 tuổi, điểm [7.5, 8.0, 7.0]

# Thêm học sinh 1
new_student1 = ['Võ Thị Giang', 17, [8.0, 8.5, 9.0]]
students.append(new_student1)

# Thêm học sinh 2
new_student2 = ['Đặng Văn Hùng', 16, [7.5, 8.0, 7.0]]
students.append(new_student2)

print("=== SAU KHI THÊM HỌC SINH MỚI ===")
for i, student in enumerate(students):
    name, age, grades = student
    print(f"{i+1}. {name} ({age} tuổi) - Điểm: {grades}")

print(f"\nTổng số học sinh hiện tại: {len(students)}")
```

Bài tập 2: Sử dụng `insert()` để thêm học sinh vào vị trí cụ thể

In [ ]:
```python
# TODO 2: Thêm học sinh mới vào vị trí thứ 3 (index 2)
# 'Bùi Thị Lan', 17 tuổi, điểm [9.0, 8.0, 8.5]

new_student = ['Bùi Thị Lan', 17, [9.0, 8.0, 8.5]]
students.insert(2, new_student)

print("=== SAU KHI INSERT HỌC SINH ===")
for i, student in enumerate(students):
    name, age, grades = student
    print(f"{i+1}. {name} ({age} tuổi) - Điểm: {grades}")
```

Phần 3: Tính điểm trung bình

Bài tập 3: Sử dụng List Comprehension để tính điểm trung bình

In [ ]:
```python
# TODO 3: Tạo list chứa điểm trung bình của tất cả học sinh
# Sử dụng List Comprehension

# Cách 1: List comprehension
averages = [sum(student[2])/len(student[2]) for student in students]

print("=== ĐIỂM TRUNG BÌNH CÁC HỌC SINH ===")
for i, student in enumerate(students):
    name = student[0]
    avg = averages[i]
    print(f"{name}: {avg:.2f}")
```

```
# TODO 4: Tính điểm trung bình của cả lớp
class_average = sum(averages) / len(averages)
print(f"\nĐiểm trung bình cả lớp: {class_average:.2f}")
```

Bài tập 4: Thêm điểm trung bình vào thông tin học sinh

In [ ]:
```
# TODO 5: Thêm điểm trung bình vào cuối mỗi record học sinh
# Biến đổi từ [tên, tuổi, [điểm]] thành [tên, tuổi, [điểm], điểm_tb]

for i, student in enumerate(students):
    # Tính điểm trung bình của học sinh này
    grades = student[2]
    avg = sum(grades) / len(grades)

    # Thêm điểm trung bình vào cuối
    student.append(avg)

print("=== HỌC SINH VỚI ĐIỂM TRUNG BÌNH ===")
for i, student in enumerate(students):
    name, age, grades, avg = student
    print(f"{i+1}. {name} ({age} tuổi) - Điểm: {grades} - TB: {avg:.2f}")
```

Phần 4: Lọc và phân loại học sinh

Bài tập 5: Lọc học sinh theo tiêu chí khác nhau

In [ ]:
```
# TODO 6: Lọc học sinh giỏi (điểm TB >= 8.0)
# Sử dụng List Comprehension với điều kiện

excellent_students = [student for student in students if student[3] >= 8.0]

print("=== HỌC SINH GIỎI (TB >= 8.0) ===")
for student in excellent_students:
    name, age, grades, avg = student
    print(f"- {name}: {avg:.2f}")

print(f"Số học sinh giỏi: {len(excellent_students)}")
```

In [ ]:
```
# TODO 7: Lọc học sinh theo tuổi và điểm

# Học sinh 16 tuổi
age_16_students = [student for student in students if student[1] == 16]

# Học sinh có điểm Toán >= 8.0
good_math_students = [student for student in students if student[2][0] >= 8.0]

# Học sinh có ít nhất 1 môn điểm 9 trở lên
high_achievers = [student for student in students if max(student[2]) >= 9.0]

print("=== THỐNG KÊ PHÂN LOẠI ===")
print(f"Học sinh 16 tuổi: {len(age_16_students)}")
```

```
print(f"Học sinh giỏi Toán: {len(good_math_students)}")
print(f"Học sinh có điểm 9+: {len(high_achievers)}")
```

Bài tập 6: Phân loại học lực

In [ ]:
```
# TODO 8: Phân loại học sinh theo điểm trung bình
# Sử dụng if/elif/else đơn giản

print("=== HỌC SINH VỚI XẾP LOẠI ===")
for i, student in enumerate(students):
    name, age, grades, avg = student

    # Phân loại theo điểm TB
    if avg >= 9.0:
        classification = "Xuất sắc"
    elif avg >= 8.0:
        classification = "Giỏi"
    elif avg >= 6.5:
        classification = "Khá"
    elif avg >= 5.0:
        classification = "Trung bình"
    else:
        classification = "Yếu"

    # Thêm xếp loại vào thông tin học sinh
    student.append(classification)

    print(f"{i+1}. {name} - TB: {avg:.2f} - Xếp loại: {classification}")
```

In [ ]:
```
# TODO 9: Đếm số học sinh theo từng loại
# Sử dụng count() method

classifications = [student[4] for student in students]  # Lấy tất cả xếp loại

xuất_sắc = classifications.count("Xuất sắc")
giỏi = classifications.count("Giỏi")
khá = classifications.count("Khá")
trung_bình = classifications.count("Trung bình")
yếu = classifications.count("Yếu")

print("=== THỐNG KÊ XẾP LOẠI ===")
print(f"Xuất sắc: {xuất_sắc} học sinh")
print(f"Giỏi: {giỏi} học sinh")
print(f"Khá: {khá} học sinh")
print(f"Trung bình: {trung_bình} học sinh")
print(f"Yếu: {yếu} học sinh")
```

Phần 5: Sắp xếp dữ liệu

Bài tập 7: Sắp xếp học sinh theo các tiêu chí khác nhau

In [ ]:
```
# TODO 10: Sắp xếp học sinh theo điểm trung bình (cao -> thấp)
# Sử dụng sorted() với vòng lặp đơn giản
```

```
# Tạo bản copy để không làm thay đổi list gốc
students_copy = students.copy()

# Sắp xếp bằng bubble sort đơn giản (để không dùng lambda)
n = len(students_copy)
for i in range(n):
    for j in range(0, n-i-1):
        # So sánh điểm TB (index 3) của 2 học sinh liền kề
        if students_copy[j][3] < students_copy[j+1][3]:  # Sắp xếp giảm dần
            students_copy[j], students_copy[j+1] = students_copy[j+1], students_cop

print("=== HỌC SINH THEO ĐIỂM TB (CAO -> THẤP) ===")
for i, student in enumerate(students_copy):
    name, age, grades, avg, classification = student
    print(f"{i+1}. {name}: {avg:.2f} ({classification})")
```

In [ ]:
```
# TODO 11: Sắp xếp theo tên (A -> Z)
students_by_name = students.copy()

# Sắp xếp theo tên bằng bubble sort
n = len(students_by_name)
for i in range(n):
    for j in range(0, n-i-1):
        # So sánh tên (index 0) của 2 học sinh liền kề
        if students_by_name[j][0] > students_by_name[j+1][0]:  # Sắp xếp tăng dần
            students_by_name[j], students_by_name[j+1] = students_by_name[j+1], stu

print("=== HỌC SINH THEO TÊN (A -> Z) ===")
for i, student in enumerate(students_by_name):
    name, age, grades, avg, classification = student
    print(f"{i+1}. {name} ({age} tuổi) - {avg:.2f}")
```

In [ ]:
```
# TODO 12: Sắp xếp theo điểm Toán (cao -> thấp)
students_by_math = students.copy()

# Sắp xếp theo điểm Toán
n = len(students_by_math)
for i in range(n):
    for j in range(0, n-i-1):
        # So sánh điểm Toán (grades[0]) của 2 học sinh liền kề
        if students_by_math[j][2][0] < students_by_math[j+1][2][0]:  # Sắp xếp giảm
            students_by_math[j], students_by_math[j+1] = students_by_math[j+1], stu

print("=== HỌC SINH THEO ĐIỂM TOÁN (CAO -> THẤP) ===")
for i, student in enumerate(students_by_math):
    name, age, grades, avg, classification = student
    math_score = grades[0]
    print(f"{i+1}. {name}: Toán {math_score} - TB {avg:.2f}")
```

Phần 6: Xóa và cập nhật dữ liệu

Bài tập 8: Xóa học sinh

In [ ]:
```python
# TODO 13: Tìm và xóa học sinh có điểm trung bình thấp nhất

# Tìm điểm TB thấp nhất
min_average = students[0][3]  # Khởi tạo với điểm TB của học sinh đầu tiên
for student in students:
    if student[3] < min_average:
        min_average = student[3]

print(f"Điểm trung bình thấp nhất: {min_average:.2f}")

# Tìm học sinh có điểm TB thấp nhất
weakest_student = None
for student in students:
    if student[3] == min_average:
        weakest_student = student
        break

print(f"Học sinh có điểm thấp nhất: {weakest_student[0]}")

# Xóa học sinh này khỏi danh sách bằng remove()
students.remove(weakest_student)

print(f"\nSau khi xóa, còn {len(students)} học sinh")
```

In [ ]:
```python
# TODO 14: Xóa học sinh ở vị trí cuối bằng pop()

print("Trước khi xóa:")
for i, student in enumerate(students):
    print(f"{i+1}. {student[0]}")

# Xóa học sinh cuối cùng bằng pop()
removed_student = students.pop()

print(f"\nĐã xóa: {removed_student[0]}")
print(f"Còn lại {len(students)} học sinh")
```

Bài tập 9: Cập nhật điểm học sinh

In [ ]:
```python
# TODO 15: Cập nhật điểm cho học sinh đầu tiên
# Thay đổi điểm Hóa của học sinh đầu tiên thành 9.5

first_student = students[0]
print(f"Trước cập nhật: {first_student[0]} - Điểm: {first_student[2]}")

# Thay đổi điểm Hóa (index 2) thành 9.5
first_student[2][2] = 9.5

# Tính lại điểm trung bình
new_grades = first_student[2]
new_average = sum(new_grades) / len(new_grades)
first_student[3] = new_average  # Cập nhật điểm TB

# Cập nhật lại xếp loại
if new_average >= 9.0:
```

```
    new_classification = "Xuất sắc"
elif new_average >= 8.0:
    new_classification = "Giỏi"
elif new_average >= 6.5:
    new_classification = "Khá"
elif new_average >= 5.0:
    new_classification = "Trung bình"
else:
    new_classification = "Yếu"

first_student[4] = new_classification

print(f"Sau cập nhật: {first_student[0]} - Điểm: {first_student[2]} - TB: {new_aver
```

Phần 7: Tổng hợp và báo cáo

Bài tập 10: Tạo báo cáo tổng hợp

In [ ]:
```
# TODO 16: Tạo báo cáo tổng hợp lớp học

print("="*50)
print("         BÁO CÁO TỔNG HỢP LỚP HỌC")
print("="*50)

# Thông tin cơ bản
total_students = len(students)
print(f"Tổng số học sinh: {total_students}")

# Điểm trung bình
all_averages = [student[3] for student in students]
class_avg = sum(all_averages) / len(all_averages)

# Tìm điểm cao nhất và thấp nhất
highest_avg = all_averages[0]
lowest_avg = all_averages[0]
for avg in all_averages:
    if avg > highest_avg:
        highest_avg = avg
    if avg < lowest_avg:
        lowest_avg = avg

print(f"\nĐIỂM SỐ:")
print(f"- Điểm TB lớp: {class_avg:.2f}")
print(f"- Điểm cao nhất: {highest_avg:.2f}")
print(f"- Điểm thấp nhất: {lowest_avg:.2f}")

# Thống kê theo môn
math_scores = [student[2][0] for student in students]
physics_scores = [student[2][1] for student in students]
chemistry_scores = [student[2][2] for student in students]

print(f"\nĐIỂM TRUNG BÌNH THEO MÔN:")
print(f"- Toán: {sum(math_scores)/len(math_scores):.2f}")
print(f"- Lý: {sum(physics_scores)/len(physics_scores):.2f}")
```

```
print(f"- Hóa: {sum(chemistry_scores)/len(chemistry_scores):.2f}")

# Thống kê xếp loại
classifications = [student[4] for student in students]
print(f"\nTHỐNG KÊ XẾP LOẠI:")
for grade in ['Xuất sắc', 'Giỏi', 'Khá', 'Trung bình', 'Yếu']:
    count = classifications.count(grade)
    percentage = (count / total_students) * 100
    print(f"- {grade}: {count} học sinh ({percentage:.1f}%)")

print("="*50)
```

Phần 8: Thử thách nâng cao

Bài tập 11: Tìm kiếm và thống kê nâng cao

In [ ]:
```
# TODO 17: Tìm học sinh có sự tiến bộ đều (điểm tăng dần Toán < Lý < Hóa)

progressive_students = []
for student in students:
    grades = student[2]
    # Kiểm tra xem điểm có tăng dần không
    if grades[0] < grades[1] < grades[2]:
        progressive_students.append(student)

print("=== HỌC SINH CÓ SỰ TIẾN BỘ ĐỀU ===")
for student in progressive_students:
    name, age, grades, avg, classification = student
    print(f"- {name}: {grades[0]} -> {grades[1]} -> {grades[2]}")

print(f"Có {len(progressive_students)} học sinh tiến bộ đều qua các môn")
```

In [ ]:
```
# TODO 18: Tìm học sinh có điểm chênh lệch lớn giữa các môn

print("=== HỌC SINH CÓ ĐIỂM CHÊNH LỆCH LỚN ===")
for student in students:
    name, age, grades, avg, classification = student

    # Tìm điểm cao nhất và thấp nhất
    max_score = grades[0]
    min_score = grades[0]

    for score in grades:
        if score > max_score:
            max_score = score
        if score < min_score:
            min_score = score

    difference = max_score - min_score

    # Nếu chênh lệch >= 1.5 điểm
    if difference >= 1.5:
        print(f"- {name}: Chênh lệch {difference:.1f} điểm (từ {min_score} đến {max
```

Tổng kết

Trong bài tập này, chúng ta đã thực hành:

List Methods đã sử dụng:

- `append()` : Thêm phần tử vào cuối list
- `insert()` : Thêm phần tử vào vị trí cụ thể
- `remove()` : Xóa phần tử theo giá trị
- `pop()` : Xóa và trả về phần tử cuối
- `count()` : Đếm số lần xuất hiện của phần tử

List Comprehensions:

- Tạo list mới từ list cũ với điều kiện
- Tính toán trên từng phần tử
- Lọc dữ liệu theo tiêu chí

Nested Lists:

- Truy cập phần tử trong list lồng nhau
- Xử lý dữ liệu phức tạp
- Quản lý thông tin có cấu trúc

Sorting và Filtering:

- Sắp xếp bằng bubble sort
- Lọc dữ liệu theo nhiều tiêu chí
- So sánh và tìm kiếm

Bài tập này giúp các bạn làm quen với việc quản lý dữ liệu thực tế bằng Python!