Bài 8: List Methods và List Comprehensions

Tuần 27 - Thứ 3 (09:00-12:00)

Mục tiêu học tập

- Hiểu và sử dụng các phương thức quan trọng của List
- Làm quen với List Comprehensions cơ bản
- Hiểu và thao tác với Nested Lists (List lồng nhau)

---

1. Ôn tập nhanh List cơ bản

Từ buổi trước chúng ta đã học:

- Tạo list: `my_list = [1, 2, 3]`
- Truy cập phần tử: `my_list[0]`
- Slicing: `my_list[1:3]`

Hôm nay chúng ta sẽ học cách thay đổi và xử lý list một cách linh hoạt hơn.

In [ ]:
```
# TODO 1: Tạo một list tên 'fruits' chứa 3 loại trái cây bạn thích
fruits =

print("List trái cây của tôi:", fruits)
print("Trái cây đầu tiên:", fruits[0])
```

2. List Methods - Các phương thức quan trọng

2.1 Thêm phần tử vào List

`append()` - Thêm 1 phần tử vào cuối list

In [ ]:
```
# Ví dụ append()
numbers = [1, 2, 3]
print("Trước khi append:", numbers)

numbers.append(4)
print("Sau khi append(4):", numbers)

numbers.append(5)
print("Sau khi append(5):", numbers)
```

In [ ]:
```
# TODO 2: Thêm 2 trái cây mới vào list 'fruits' ở trên bằng append()
# Rồi in ra list để xem kết quả
```

`insert()` - Thêm phần tử vào vị trí cụ thể

In [ ]:
```
# Ví dụ insert()
colors = ['đỏ', 'xanh', 'vàng']
print("Trước khi insert:", colors)

# insert(vị_trí, giá_trị)
colors.insert(1, 'cam')  # Thêm 'cam' vào vị trí thứ 1
print("Sau khi insert('cam' vào vị trí 1):", colors)

colors.insert(0, 'tím')  # Thêm 'tím' vào đầu list
print("Sau khi insert('tím' vào vị trí 0):", colors)
```

In [ ]:
```
# TODO 3: Tạo list học sinh và thử insert
students = ['An', 'Bình', 'Chi']
print("Danh sách ban đầu:", students)

# Thêm 'Dung' vào vị trí thứ 2

# Thêm 'Em' vào đầu danh sách

print("Danh sách sau khi thêm:", students)
```

2.2 Xóa phần tử khỏi List

`remove()` - Xóa phần tử theo giá trị

In [4]:
```
# Ví dụ remove()
animals = ['mèo', 'chó', 'gà', 'chó', 'vịt']
print("Trước khi remove:", animals)

animals.remove('chó')  # Chỉ xóa chó đầu tiên tìm thấy
print("Sau khi remove('chó'):", animals)

animals.remove('gà')
print("Sau khi remove('gà'):", animals)
```

```
Trước khi remove: ['mèo', 'chó', 'gà', 'chó', 'vịt']
Sau khi remove('chó'): ['mèo', 'gà', 'chó', 'vịt']
Sau khi remove('gà'): ['mèo', 'chó', 'vịt']
```

`pop()` - Xóa phần tử theo vị trí và trả về giá trị đó

In [1]:
```
# Ví dụ pop()
foods = ['phở', 'cơm', 'bánh mì', 'bún']
print("Trước khi pop:", foods)

# pop() không có tham số = xóa phần tử cuối
last_food = foods.pop()
```

```
print(f"Đã xóa: {last_food}")
print("Sau khi pop():", foods)

# pop(vị_trí)
first_food = foods.pop(0)
print(f"Đã xóa: {first_food}")
print("Sau khi pop(0):", foods)
```

```
Trước khi pop: ['phở', 'cơm', 'bánh mì', 'bún']
Đã xóa: bún
Sau khi pop(): ['phở', 'cơm', 'bánh mì']
Đã xóa: phở
Sau khi pop(0): ['cơm', 'bánh mì']
```

In [ ]:
```
# TODO 4: Thực hành remove và pop
subjects = ['Toán', 'Lý', 'Hóa', 'Sinh', 'Văn', 'Sử']
print("Môn học ban đầu:", subjects)

# Xóa môn 'Hóa' bằng remove

# Xóa môn cuối cùng bằng pop và lưu vào biến removed_subject

print("Môn học còn lại:", subjects)
print("Môn vừa xóa:", removed_subject)
```

2.3 Các phương thức hữu ích khác

`count()` - Đếm số lần xuất hiện

`index()` - Tìm vị trí của phần tử

`sort()` - Sắp xếp list

`reverse()` - Đảo ngược list

In [5]:
```
# Ví dụ các phương thức khác
grades = [8, 7, 9, 7, 10, 7, 8]
print("Điểm số:", grades)

# count() - đếm
count_7 = grades.count(7)
print(f"Số lần xuất hiện điểm 7: {count_7}")

# index() - tìm vị trí
pos_10 = grades.index(10)
print(f"Vị trí của điểm 10: {pos_10}")

# sort() - sắp xếp (thay đổi list gốc)
grades.sort()
print("Sau khi sắp xếp:", grades)

# reverse() - đảo ngược
```

```
grades.reverse()
print("Sau khi đảo ngược:", grades)
```

```
Điểm số: [8, 7, 9, 7, 10, 7, 8]
Số lần xuất hiện điểm 7: 3
Vị trí của điểm 10: 4
Sau khi sắp xếp: [7, 7, 7, 8, 8, 9, 10]
Sau khi đảo ngược: [10, 9, 8, 8, 7, 7, 7]
```

In [ ]:
```
# TODO 5: Thực hành với các phương thức
temperatures = [25, 30, 28, 25, 32, 28, 25, 35]
print("Nhiệt độ trong tuần:", temperatures)

# Đếm xem có bao nhiêu ngày nhiệt độ 25 độ
days_25 =
print(f"Số ngày nhiệt độ 25 độ: {days_25}")

# Tìm vị trí của nhiệt độ cao nhất (35 độ)
pos_max =
print(f"Ngày có nhiệt độ cao nhất là ngày thứ: {pos_max + 1}")

# Sắp xếp nhiệt độ từ thấp đến cao
temperatures.sort()
print("Nhiệt độ đã sắp xếp:", temperatures)
```

3. List Comprehensions cơ bản

List Comprehension là cách viết ngắn gọn để tạo list mới từ list cũ.

Cú pháp cơ bản: `[biểu_thức for phần_tử in list_cũ]`

So sánh cách viết truyền thống và List Comprehension

In [ ]:
```
# Ví dụ 1: Tạo list bình phương
numbers = [1, 2, 3, 4, 5]

# Cách viết truyền thống với for loop
squares_old = []
for num in numbers:
    squares_old.append(num ** 2)
print("Cách cũ:", squares_old)

# Cách viết với List Comprehension
squares_new = [num ** 2 for num in numbers]
print("Cách mới:", squares_new)
```

In [ ]:
```
# Ví dụ 2: Chuyển đổi nhiệt độ
celsius = [0, 20, 30, 40]

# Chuyển từ độ C sang độ F: F = C * 9/5 + 32
fahrenheit = [c * 9/5 + 32 for c in celsius]
print(f"Celsius: {celsius}")
print(f"Fahrenheit: {fahrenheit}")
```

In [ ]:
```python
# TODO 6: Thực hành List Comprehension
prices = [100, 250, 80, 150, 300]
print("Giá gốc:", prices)

# Tạo list giá sau khi giảm 20% bằng List Comprehension
sale_prices =
print("Giá sau giảm 20%:", sale_prices)

# Tạo list tên học sinh viết hoa
names = ['an', 'binh', 'chi', 'dung']
upper_names =
print("Tên viết hoa:", upper_names)
```

List Comprehension với điều kiện

Cú pháp: `[biểu_thức for phần_tử in list_cũ if điều_kiện]`

In [5]:
```python
# Ví dụ: Lọc số chẵn và nhân đôi
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Lấy các số chẵn và nhân đôi
even_doubled = [num * 2 for num in numbers if num % 2 == 0]
print("Số chẵn nhân đôi:", even_doubled)

# Lấy các từ có độ dài > 3
words = ['mèo', 'chó', 'gà', 'vịt', 'bò', 'trâu']
long_words = [word for word in words if len(word) > 3]
print("Từ có độ dài > 3:", long_words)
```

```
Số chẵn nhân đôi: [4, 8, 12, 16, 20]
Từ có độ dài > 3: ['trâu']
```

In [ ]:
```python
# TODO 7: List Comprehension với điều kiện
scores = [95, 67, 82, 45, 78, 90, 55, 88]
print("Điểm số:", scores)

# Tạo list chứa các điểm >= 80 (điểm giỏi)
good_scores =
print("Điểm giỏi (>= 80):", good_scores)

# Tạo list chứa bình phương của các số lẻ
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9]
odd_squares =
print("Bình phương số lẻ:", odd_squares)
```

4. Nested Lists (List lồng nhau)

Nested List là list chứa các list khác bên trong. Rất hữu ích để lưu trữ dữ liệu dạng bảng, ma trận.

4.1 Tạo và truy cập Nested List

In [ ]:
```python
# Ví dụ: Bảng điểm của 3 học sinh, mỗi học sinh có 4 môn
student_grades = [
    [8, 9, 7, 8],    # An: Toán, Lý, Hóa, Văn
    [7, 8, 9, 7],    # Bình
    [9, 7, 8, 9]     # Chi
]

print("Bảng điểm:")
print(student_grades)

# Truy cập điểm của học sinh đầu tiên (An)
an_grades = student_grades[0]
print("\nĐiểm của An:", an_grades)

# Truy cập điểm Toán của An
an_math = student_grades[0][0]
print("Điểm Toán của An:", an_math)

# Truy cập điểm Văn của Bình
binh_literature = student_grades[1][3]
print("Điểm Văn của Bình:", binh_literature)
```

In [ ]:
```python
# TODO 8: Thực hành với Nested List
# Tạo nested list lưu thông tin sản phẩm: [tên, giá, số_lượng]
products = [
    ['Táo', 15000, 10],
    ['Cam', 20000, 8],
    ['Chuối', 12000, 15]
]

print("Danh sách sản phẩm:", products)

# Lấy tên sản phẩm đầu tiên
first_product_name =
print("Sản phẩm đầu tiên:", first_product_name)

# Lấy giá của cam (sản phẩm thứ 2)
orange_price =
print("Giá cam:", orange_price)

# Lấy số lượng chuối (sản phẩm thứ 3)
banana_quantity =
print("Số lượng chuối:", banana_quantity)
```

4.2 Duyệt qua Nested List với vòng lặp

In [ ]:
```python
# Ví dụ: In ra thông tin tất cả sản phẩm
products = [
    ['Táo', 15000, 10],
    ['Cam', 20000, 8],
    ['Chuối', 12000, 15]
]

print("Thông tin sản phẩm:")
```

```
for i in range(len(products)):
    name = products[i][0]
    price = products[i][1]
    quantity = products[i][2]
    print(f"{i+1}. {name}: {price}đ - Còn {quantity} cái")

print("\nCách viết khác:")
for product in products:
    name, price, quantity = product  # Unpacking
    print(f"- {name}: {price}đ - Còn {quantity} cái")
```

4.3 Thay đổi giá trị trong Nested List

In [ ]:
```
# Ví dụ: Cập nhật thông tin sản phẩm
products = [
    ['Táo', 15000, 10],
    ['Cam', 20000, 8],
    ['Chuối', 12000, 15]
]

print("Trước khi cập nhật:", products)

# Tăng giá táo lên 17000
products[0][1] = 17000

# Giảm số lượng cam đi 3
products[1][2] -= 3

# Thêm sản phẩm mới
products.append(['Xoài', 25000, 5])

print("Sau khi cập nhật:", products)
```

In [ ]:
```
# TODO 9: Bài tập tổng hợp Nested List
# Tạo nested list lưu điểm 3 học sinh, mỗi em 3 môn
class_grades = [
    ['An', [8, 9, 7]],      # [Tên, [Toán, Lý, Hóa]]
    ['Bình', [7, 8, 9]],
    ['Chi', [9, 7, 8]]
]

print("Bảng điểm lớp:")
# Tính và in điểm trung bình của từng học sinh
for student in class_grades:
    name = student[0]
    grades = student[1]
    average = sum(grades) / len(grades)
    print(f"{name}: {grades} -> Trung bình: {average:.1f}")

# TODO: Tìm học sinh có điểm trung bình cao nhất
best_student = ""
highest_avg = 0

# Viết code để tìm học sinh có điểm TB cao nhất
```

```
print(f"\nHọc sinh giỏi nhất: {best_student} với điểm TB {highest_avg:.1f}")
```

5. Bài tập thực tế: Hệ thống Shopping Cart

Áp dụng kiến thức vừa học để xây dựng một hệ thống giỏ hàng đơn giản.

5.1 Xây dựng Shopping Cart cơ bản

In [ ]:
```
# Hệ thống Shopping Cart
# Giỏ hàng sẽ là nested list: [tên_sản_phẩm, giá, số_lượng]
shopping_cart = []
available_products = [
    ['Áo thun', 150000],
    ['Quần jean', 250000],
    ['Giày thể thao', 400000],
    ['Túi xách', 200000]
]

def show_products():
    """Hiển thị danh sách sản phẩm có sẵn"""
    print("\n=== DANH SÁCH SẢN PHẨM ===")
    for i, product in enumerate(available_products):
        name, price = product
        print(f"{i+1}. {name}: {price:,}đ")

def add_to_cart(product_name, price, quantity):
    """Thêm sản phẩm vào giỏ hàng"""
    # Kiểm tra xem sản phẩm đã có trong giỏ chưa
    for item in shopping_cart:
        if item[0] == product_name:
            item[2] += quantity  # Tăng số lượng
            print(f"Đã tăng số lượng {product_name} thành {item[2]}")
            return

    # Nếu chưa có, thêm mới
    shopping_cart.append([product_name, price, quantity])
    print(f"Đã thêm {quantity} {product_name} vào giỏ hàng")

# Demo
show_products()

# Thêm một số sản phẩm vào giỏ
add_to_cart('Áo thun', 150000, 2)
add_to_cart('Giày thể thao', 400000, 1)
add_to_cart('Áo thun', 150000, 1)  # Thêm áo thun lần nữa

print("\nGiỏ hàng hiện tại:", shopping_cart)
```

In [ ]:
```
# TODO 11: Hoàn thiện Shopping Cart
def show_cart():
    """Hiển thị giỏ hàng chi tiết"""
    if not shopping_cart:
```

```
        print("Giỏ hàng trống!")
        return

    print("\n=== GIỎ HÀNG CỦA BẠN ===")
    total = 0
    # TODO: Viết code hiển thị từng sản phẩm và tính tổng tiền
    for item in shopping_cart:
        name, price, quantity = item
        subtotal = price * quantity
        total += subtotal
        print(f"- {name}: {price:,}đ x {quantity} = {subtotal:,}đ")

    print(f"\nTONG CỘNG: {total:,}đ")
    return total

def remove_from_cart(product_name):
    """Xóa sản phẩm khỏi giỏ hàng"""
    # TODO: Viết code xóa sản phẩm
    for i, item in enumerate(shopping_cart):
        if item[0] == product_name:
            removed_item = shopping_cart.pop(i)
            print(f"Đã xóa {removed_item[0]} khỏi giỏ hàng")
            return
    print(f"Không tìm thấy {product_name} trong giỏ hàng")

# Test các function
show_cart()
remove_from_cart('Áo thun')
show_cart()
```

5.2 Nâng cao với List Comprehensions

In [ ]:

```
# Sử dụng List Comprehensions cho Shopping Cart
shopping_cart = [
    ['Áo thun', 150000, 2],
    ['Quần jean', 250000, 1],
    ['Giày thể thao', 400000, 1],
    ['Túi xách', 200000, 3]
]

# Tính tổng tiền từng sản phẩm
subtotals = [price * quantity for name, price, quantity in shopping_cart]
print("Tổng tiền từng sản phẩm:", subtotals)

# Lấy tên các sản phẩm đắt tiền (>= 200,000)
expensive_items = [name for name, price, quantity in shopping_cart if price >= 2000
print("Sản phẩm đắt tiền:", expensive_items)

# Tạo list thông báo cho từng sản phẩm
notifications = [f"{name}: {quantity} cái" for name, price, quantity in shopping_ca
print("Thông báo:")
for notification in notifications:
    print(f"- {notification}")

# Tổng tiền toàn bộ giỏ hàng
```

```
total_amount = sum([price * quantity for name, price, quantity in shopping_cart])
print(f"\nTổng tiền: {total_amount:,}đ")
```

In [ ]:
```
# TODO 12: Thực hành với Shopping Cart
sales_data = [
    ['Áo thun', 150000, 5, 'S'],    # [tên, giá, số_lượng, size]
    ['Áo thun', 150000, 3, 'M'],
    ['Áo thun', 150000, 2, 'L'],
    ['Quần jean', 250000, 4, 'M'],
    ['Giày thể thao', 400000, 2, '42']
]

# Sử dụng List Comprehension để:

# 1. Lấy tất cả áo thun (bất kể size)
tshirts = [item for item in sales_data if item[0] == 'Áo thun']
print("Các áo thun:", tshirts)

# 2. Tính tổng doanh thu từ áo thun
tshirt_revenue = sum([price * quantity for name, price, quantity, size in sales_dat
print(f"Doanh thu từ áo thun: {tshirt_revenue:,}đ")

# 3. Lấy danh sách size có sẵn
available_sizes = [size for name, price, quantity, size in sales_data]
unique_sizes = list(set(available_sizes))  # Loại bỏ trùng lặp
print("Size có sẵn:", unique_sizes)
```

6. So sánh List với các Data Structures khác

Để chuẩn bị cho những buổi học tiếp theo, chúng ta sẽ tìm hiểu sơ qua sự khác biệt giữa List và các cấu trúc dữ liệu khác.

6.1 List vs Tuple vs Set vs Dictionary

In [ ]:
```
# So sánh các loại data structures
print("=== SO SÁNH CÁC DATA STRUCTURES ===")

# LIST - Có thể thay đổi, có thứ tự, cho phép trùng lặp
my_list = [1, 2, 2, 3, 4]
print(f"List: {my_list}")
my_list.append(5)  # Có thể thay đổi
print(f"Sau khi append: {my_list}")
print(f"Truy cập phần tử: my_list[0] = {my_list[0]}")

# TUPLE - Không thể thay đổi, có thứ tự, cho phép trùng lặp
my_tuple = (1, 2, 2, 3, 4)
print(f"\nTuple: {my_tuple}")
# my_tuple.append(5)  # ERROR! Không thể thay đổi
print(f"Truy cập phần tử: my_tuple[0] = {my_tuple[0]}")

# SET - Có thể thay đổi, không có thứ tự, KHÔNG cho phép trùng lặp
my_set = {1, 2, 2, 3, 4}  # Số 2 trùng sẽ chỉ giữ lại 1 cái
print(f"\nSet: {my_set}")
```

```
my_set.add(5)  # Có thể thay đổi
print(f"Sau khi add: {my_set}")
# print(my_set[0])  # ERROR! Không có index

# DICTIONARY - Có thể thay đổi, có thứ tự (Python 3.7+), key không trùng
my_dict = {'name': 'An', 'age': 20, 'city': 'HN'}
print(f"\nDictionary: {my_dict}")
my_dict['school'] = 'ABC University'  # Có thể thay đổi
print(f"Sau khi thêm: {my_dict}")
print(f"Truy cập: my_dict['name'] = {my_dict['name']}")
```

6.2 Khi nào dùng cái gì?

In [ ]:

```
# Hướng dẫn chọn data structure phù hợp
print("=== HƯỚNG DẪN CHỌN DATA STRUCTURE ===")

# Dùng LIST khi:
print("\n📝 Dùng LIST khi:")
print("- Cần lưu trữ nhiều giá trị có thứ tự")
print("- Cần thay đổi dữ liệu (thêm, xóa, sửa)")
print("- Cho phép giá trị trùng lặp")
print("- Ví dụ: Danh sách điểm số, giỏ hàng, todo list")

scores = [8, 9, 7, 8, 10]  # Có thể có điểm trùng
todo_list = ['Học bài', 'Làm bài tập', 'Đi chơi']

# Dùng TUPLE khi:
print("\n📌 Dùng TUPLE khi:")
print("- Cần lưu dữ liệu không thay đổi")
print("- Thường dùng cho tọa độ, RGB color, config")
print("- Ví dụ: Tọa độ điểm, màu sắc, thông tin cố định")

point = (10, 20)  # Tọa độ x, y
rgb_color = (255, 128, 0)  # Màu cam

# Dùng SET khi:
print("\n🔍 Dùng SET khi:")
print("- Cần loại bỏ giá trị trùng lặp")
print("- Cần kiểm tra membership nhanh")
print("- Ví dụ: Danh sách unique users, tags, categories")

unique_visitors = {1001, 1002, 1003, 1001}  # Tự động loại bỏ ID trùng
print(f"Unique visitors: {unique_visitors}")

# Dùng DICTIONARY khi:
print("\n🗂️ Dùng DICTIONARY khi:")
print("- Cần lưu trữ dữ liệu dạng key-value")
print("- Cần truy cập nhanh theo key")
print("- Ví dụ: Thông tin user, cấu hình, mapping")

user_info = {'id': 1001, 'name': 'An', 'email': 'an@email.com'}
grade_mapping = {'A': 90, 'B': 80, 'C': 70}
```

In [ ]:
```python
# TODO 13: Thực hành chọn data structure
# Cho các tình huống sau, em hãy chọn data structure phù hợp và giải thích

print("=== BÀI TẬP CHỌN DATA STRUCTURE ===")

# Tình huống 1: Lưu danh sách môn học của 1 học sinh
# Cần thêm/xóa môn, có thể có môn trùng (học lại)
student_subjects = ['Toán', 'Lý', 'Hóa', 'Toán']  # LIST
print(f"1. Môn học: {student_subjects} -> Dùng LIST")

# Tình huống 2: Lưu tọa độ của một điểm trên bản đồ
# Không thay đổi sau khi xác định
location = (21.0285, 105.8542)  # TUPLE
print(f"2. Tọa độ Hà Nội: {location} -> Dùng TUPLE")

# Tình huống 3: Lưu danh sách các thành phố đã đi du lịch
# Không quan tâm thứ tự, không có thành phố trùng
visited_cities = {'Hà Nội', 'TP.HCM', 'Đà Nẵng', 'Hà Nội'}  # SET
print(f"3. Thành phố đã đi: {visited_cities} -> Dùng SET")

# Tình huống 4: Lưu thông tin chi tiết của sản phẩm
# Cần truy cập theo tên thuộc tính
product_info = {'name': 'iPhone', 'price': 20000000, 'color': 'Đen'}  # DICT
print(f"4. Thông tin sản phẩm: {product_info} -> Dùng DICT")

# TODO: Em hãy tự tạo thêm 2 ví dụ và chọn data structure phù hợp
print("\n=== BÀI TẬP TỰ LÀM ===")
# Ví dụ 5: _____
# Ví dụ 6: _____
```

7. Bài tập tổng hợp cuối buổi

Áp dụng tất cả kiến thức đã học trong một bài tập lớn.

In [ ]:
```python
# BÀI TẬP TỔNG HỢP: QUẢN LÝ CỬA HÀNG SÁCH
print("=== HỆ THỐNG QUẢN LÝ CỬA HÀNG SÁCH ===")

# Dữ liệu sách: [tên_sách, tác_giả, giá, số_lượng, thể_loại]
books = [
    ['Python Programming', 'John Smith', 350000, 15, 'Công nghệ'],
    ['Dế Mèn Phiêu Lưu Ký', 'Tô Hoài', 120000, 20, 'Thiếu nhi'],
    ['Sapiens', 'Yuval Harari', 280000, 10, 'Lịch sử'],
    ['Clean Code', 'Robert Martin', 420000, 8, 'Công nghệ'],
    ['Tắt Đèn', 'Ngô Tất Tố', 150000, 25, 'Văn học'],
    ['Data Science Handbook', 'Jake VanderPlas', 480000, 5, 'Công nghệ']
]

print(f"Cửa hàng có {len(books)} loại sách")

# 1. Hiển thị tất cả sách
def show_all_books():
    print("\n=== DANH SÁCH TẤT CẢ SÁCH ===")
    for i, book in enumerate(books):
```

```
        title, author, price, stock, genre = book
        print(f"{i+1}. {title} - {author}")
        print(f"   Giá: {price:,}đ | Còn: {stock} cuốn | Thể loại: {genre}")

show_all_books()
```

In [ ]:

```
# TODO 14: Hoàn thiện hệ thống quản lý sách

# 2. Tìm sách theo thể loại
def find_books_by_genre(genre):
    """Tìm tất cả sách thuộc thể loại cụ thể"""
    result = [book for book in books if book[4] == genre]

    if result:
        print(f"\n=== SÁCH THỂ LOẠI '{genre}' ===")
        for book in result:
            title, author, price, stock, _ = book
            print(f"- {title} by {author}: {price:,}đ")
    else:
        print(f"Không tìm thấy sách thể loại '{genre}'")

    return result

# 3. Tìm sách trong tầm giá
def find_books_by_price_range(min_price, max_price):
    """Tìm sách trong khoảng giá"""
    # TODO: Viết code tìm sách có giá từ min_price đến max_price
    result = [book for book in books if min_price <= book[2] <= max_price]

    print(f"\n=== SÁCH TỪ {min_price:,}đ ĐẾN {max_price:,}đ ===")
    for book in result:
        title, author, price, stock, genre = book
        print(f"- {title}: {price:,}đ ({genre})")

    return result

# 4. Thống kê tồn kho
def inventory_stats():
    """Thống kê tình hình tồn kho"""
    # TODO: Tính tổng số sách, tổng giá trị, sách sắp hết
    total_books = sum([stock for _, _, _, stock, _ in books])
    total_value = sum([price * stock for _, _, price, stock, _ in books])
    low_stock = [book for book in books if book[3] < 10]  # Dưới 10 cuốn

    print("\n=== THỐNG KÊ TỒN KHO ===")
    print(f"Tổng số sách: {total_books} cuốn")
    print(f"Tổng giá trị: {total_value:,}đ")
    print(f"Sách sắp hết ({len(low_stock)} loại):")
    for book in low_stock:
        print(f"  - {book[0]}: {book[3]} cuốn")

# Test các function
find_books_by_genre('Công nghệ')
find_books_by_price_range(100000, 300000)
inventory_stats()
```

In [ ]:
```python
# TODO 15: Thử thách cuối buổi
# Tạo báo cáo chi tiết với List Comprehensions

print("=== BÁO CÁO CHI TIẾT ===")

# 1. Top 3 sách đắt nhất
expensive_books = sorted(books, key=lambda x: x[2], reverse=True)[:3]
print("\n📊 TOP 3 SÁCH ĐẮT NHẤT:")
for i, book in enumerate(expensive_books):
    print(f"{i+1}. {book[0]}: {book[2]:,}đ")

# 2. Các thể loại có trong cửa hàng
genres = list(set([book[4] for book in books]))
print(f"\n📚 CÁC THỂ LOẠI: {', '.join(genres)}")

# 3. Sách có tên tác giả Việt Nam (chứa ký tự Việt)
vietnamese_authors = [book for book in books if any(char in 'àáảãạâầấẩẫậăằắẳẵặèéẻẽẹ
print(f"\nVN SÁCH TÁC GIẢ VIỆT ({len(vietnamese_authors)} cuốn):")
for book in vietnamese_authors:
    print(f"- {book[0]} - {book[1]}")

# 4. Tính tỉ lệ từng thể loại
total_books = len(books)
genre_stats = {}
for genre in genres:
    count = len([book for book in books if book[4] == genre])
    percentage = (count / total_books) * 100
    genre_stats[genre] = {'count': count, 'percentage': percentage}

print("\n📈 THỐNG KÊ THEO THỂ LOẠI:")
for genre, stats in genre_stats.items():
    print(f"- {genre}: {stats['count']} cuốn ({stats['percentage']:.1f}%)")

# TODO: Em hãy thêm 2 thống kê khác bằng List Comprehension
print("\n=== THỐNG KÊ BỔ SUNG ===")
# 5. _____
# 6. _____
```

https://create.kahoot.it/share/bo-cau-hoi-trac-nghiem-python-fundamentals/1b594fae-29b2-4d78-b25e-b26a73fd6cf8

https://create.kahoot.it/share/w27-t3/7f805c97-8296-4b75-8815-7a2ff305c6f0

8. Tổng kết và chuẩn bị cho buổi sau

Kiến thức đã học hôm nay:

🔧 List Methods quan trọng:

- `append()` : Thêm phần tử vào cuối
- `insert()` : Thêm phần tử vào vị trí cụ thể
- `remove()` : Xóa phần tử theo giá trị

- `pop()`: Xóa phần tử theo vị trí
- `count()`, `index()`, `sort()`, `reverse()`

⚡ List Comprehensions:

- Cú pháp cơ bản: `[biểu_thức for phần_tử in list]`
- Với điều kiện: `[biểu_thức for phần_tử in list if điều_kiện]`
- Giúp code ngắn gọn và hiệu quả hơn

📊 Nested Lists:

- List chứa các list khác bên trong
- Truy cập: `nested_list[i][j]`
- Hữu ích cho dữ liệu dạng bảng, ma trận

🗂️ So sánh Data Structures:

- **List**: Có thể thay đổi, có thứ tự, cho phép trùng lặp
- **Tuple**: Không thể thay đổi, có thứ tự, cho phép trùng lặp
- **Set**: Có thể thay đổi, không thứ tự, không trùng lặp
- **Dictionary**: Key-value pairs, không trùng key

🎯 Chuẩn bị cho buổi chiều:

- **Bài tập**: Student Management System
- **Kỹ năng**: Sorting, filtering, List manipulation
- **Ứng dụng**: Quản lý dữ liệu thực tế

📚 Chuẩn bị cho tuần sau:

- **Thứ 4**: Dictionary cơ bản và Dictionary methods
- **Thứ 5**: Tuples và Sets chi tiết
- **Thứ 6**: Mini project Contact Manager

💡 Lời khuyên:

1. **Thực hành nhiều**: List methods cần được luyện tập thường xuyên
2. **Hiểu khi nào dùng gì**: List vs Tuple vs Set vs Dict
3. **List Comprehension**: Bắt đầu từ đơn giản rồi phức tạp dần
4. **Nested Lists**: Hữu ích cho dữ liệu thực tế như bảng Excel

🏠 Bài tập về nhà:

1. Hoàn thiện tất cả TODO trong bài học
2. Tạo 1 chương trình quản lý điểm số cá nhân
3. Thử nghiệm với các List methods khác nhau

4. Chuẩn bị cho bài Student Management buổi chiều

Chúc các bạn học tốt! 🌟