# IE221 - BÀI THỰC HÀNH 3

# Bài tập ứng dụng về các câu lệnh điều kiện và vòng lặp

Ngày 10 tháng 10 năm 2025

## Mục lục

# 1. NHẮC LẠI LÝ THUYẾT VÀ KHÁI NIỆM

## 1.1 Câu lệnh điều kiện (Conditional Statements)

Câu lệnh điều kiện cho phép chương trình kiểm tra một điều kiện và thực hiện các thao tác khác nhau tùy vào kết quả kiểm tra đó. Các dạng phổ biến:

- if: Kiểm tra điều kiện, nếu đúng thì thực hiện khối lệnh.
- if-else: Nếu điều kiện đúng thì thực hiện khối lệnh 1, nếu sai thì thực hiện khối lệnh 2.
- if-elif-else: Kiểm tra nhiều điều kiện tuần tự.
- Toán tử ba ngôi: Viết gọn if-else trên một dòng.

Cú pháp cơ bản:

```
if <điều_kiện>:
    <khối_lệnh_1>
elif <điều_kiện_khác>:
    <khối_lệnh_2>
else:
    <khối_lệnh_mặc_định>
```

Ví dụ:

```
x = 5
if x > 0:
    print("Số dương")
elif x == 0:
    print("Số không")
else:
    print("Số âm")
```

## 1.2 Cấu trúc lặp (Loop Statements)

Cho phép thực hiện lặp đi lặp lại một khối lệnh đến khi điều kiện dừng được thoả mãn. Hai loại vòng lặp chính:

- for (lặp theo dãy phần tử)
- while (lặp khi điều kiện còn đúng)

### 1.2.1 Vòng lặp for

Dùng để lặp qua một chuỗi, danh sách, hoặc một dãy số. Cú pháp:

```
for <biến> in <chuỗi_dữ_liệu>:
    <khối_lệnh>
```

Ví dụ:

```
for i in range(1, 6):
    print("Lần lặp thứ:", i)
```

Ghi chú:

- range(a, b) sinh ra các số từ a đến b-1.
- Dùng enumerate() để lấy cả chỉ số và giá trị:

```
for idx, val in enumerate(['a', 'b', 'c']):
    print(idx, val)
```

#### 1.2.2 Vòng lặp while

Lặp lại khối lệnh khi điều kiện còn đúng. Cú pháp:

```
while <điều_kiện>:
    <khối_lệnh>
```

Ví dụ:

```
count = 0
while count < 5:
    print("Giá trị:", count)
    count += 1
```

Ghi chú:

- Nếu quên cập nhật biến điều kiện, vòng lặp có thể chạy vô hạn (infinite loop).

#### 1.2.3 Từ khoá điều khiển vòng lặp

- break: Thoát khỏi vòng lặp ngay lập tức.
- continue: Bỏ qua lần lặp hiện tại, chuyển sang lần tiếp theo.
- pass: Bỏ qua

### 1.3 Vòng lặp lồng nhau (nested loop)

Vòng lặp lồng nhau thường dùng để xử lý các bài toán về ma trận, in mẫu ký tự (pattern printing), hoặc các thuật toán phức tạp hơn.

```
for i in range(3):
    for j in range(3):
        print(f"({i},{j})", end=" ")
    print()
```

## 2. BÀI TẬP DEMO TẠI LỚP

### 2.1 Demo 1: Kiểm tra số nguyên tố

**Yêu cầu**: Viết hàm kiểm tra một số nguyên tố, sử dụng vòng lặp và câu lệnh điều kiện.
**Gợi ý cách làm**:

- Số nguyên tố là số lớn hơn 1, chỉ chia hết cho 1 và chính nó.
- Kiểm tra từ 2 đến căn bậc hai của số đó.

```
def kiem_tra_nguyen_to(n):
    """Kiểm tra một số có phải là số nguyên tố không"""
    if n < 2:
        return False
    elif n == 2:
        return True
    elif
    ...

# Test
so = int(input("Nhập số cần kiểm tra: "))
ket_qua = "là số nguyên tố" if kiem_tra_nguyen_to(so) else "không là số nguyên tố"
print(f"{so} {ket_qua}")
```

### 2.2 Demo 2: Menu cơ bản với vòng lặp while

**Yêu cầu**: Xây dựng menu chọn phép toán, xử lý nhập liệu, vòng lặp while.
**Gợi ý cách làm**:

- Sử dụng vòng lặp while để hiển thị menu liên tục.

- Cho nhập phép người dùng nhập 2 số.
- Dùng if-elif để xử lý từng lựa chọn.
- Kiểm tra chia cho 0.
- Không kết thúc chương trình sau khi ra kết quả.

```
=== MÁY TÍNH ===
1. Cộng
2. Trừ
3. Nhân
4. Chia
0. Thoát
Chọn phép toán (0-4):
```

Hình 1: Mẫu menu

```
Chọn phép toán (0-4): 1
Nhập số thứ nhất: 2
Nhập số thứ hai: 6
Kết quả: 2.0 + 6.0 = 8.0

=== MÁY TÍNH ===
1. Cộng
2. Trừ
3. Nhân
4. Chia
0. Thoát
Chọn phép toán (0-4):
```

Hình 2: Minh họa sau khi ra kết quả

## 2.3 Demo 3: Game đoán số (interactive)

**Yêu cầu:** Viết game đoán số, random số trong , người chơi đoán đến khi đúng, có gợi ý và đếm số lần đoán.
**Gợi ý cách làm:**

- Sử dụng random.randint để sinh số bí mật.
- Vòng lặp while cho đến khi đoán đúng.
- Mỗi lần đoán, so sánh và đưa ra gợi ý "lớn hơn" hoặc "nhỏ hơn".

```
Đoán số từ 1 đến 100
Nhập số bạn đoán: 50
Số thực tế lớn hơn.
Nhập số bạn đoán: 70
Số thực tế nhỏ hơn.
Nhập số bạn đoán: 65
Số thực tế nhỏ hơn.
Nhập số bạn đoán: 55
Số thực tế nhỏ hơn.
Nhập số bạn đoán: 53
Đúng! Số cần đoán là 53. Bạn đã đoán 5 lần.
```

Hình 3: Minh họa kết quả

# 3. BÀI TẬP VỀ NHÀ

## 3.1 Bài 1 – Validate input và điều kiện

**Yêu cầu**: Viết hàm `read_int(prompt)` dùng while + try/except, buộc nhập đúng số nguyên.
**Gợi ý cách làm:**

- Dùng vòng lặp while để nhập liên tục.
- Dùng try/except để kiểm tra kiểu dữ liệu.

## 3.2 Bài 2 – Giai thừa (factorial)

**Yêu cầu:** Viết factorial(n) bằng cả for và while, với kiểm tra đầu vào n >= 0.
**Gợi ý cách làm:**

- Kiểm tra đầu vào n >= 0.
- Tính giai thừa bằng for và while.

## 3.3 Bài 3 – Tìm số hoàn hảo

**Yêu cầu:** Kiểm tra và liệt kê các số hoàn hảo trong một khoảng.
**Gợi ý cách làm:**

- Số hoàn hảo là số bằng tổng các ước số thực sự của nó (không tính chính nó).
- Dùng vòng lặp for để kiểm tra từng số trong khoảng.

## 3.4 Bài 4 – Quản lý điểm sinh viên

**Yêu cầu:** Xây dựng chương trình quản lý điểm có các chức năng:

- Nhập danh sách sinh viên và điểm
- Tính điểm trung bình, xếp loại
- Tìm sinh viên có điểm cao nhất, thấp nhất
- Thống kê số sinh viên theo từng loại

```
    def quan_ly_diem():
        sinh_vien = {}

        while True:
            print("\n=== HỆ THỐNG QUẢN LÝ ĐIỂM ===")
            print("1. Nhập thông tin sinh viên")
            print("2. Hiển thị bảng điểm")
            print("3. Thống kê theo xếp loại")
            print("4. Tìm sinh viên điểm cao/thấp nhất")
            print("0. Thoát")

            lua_chon = input("Chọn chức năng: ")

            if lua_chon == '0':
                break
            elif lua_chon == '1':
                # Sinh viên tự hoàn thiện
                pass
            # ... các chức năng khác
```

## 3.5 Bài 5 – Game đoán số nâng cao

Yêu cầu: Tạo game đoán số với các tính năng:

- Nhiều mức độ khó khác nhau
- Hệ thống gợi ý thông minh
- Thống kê số lần chơi và tỷ lệ thắng
- Lưu high score

## 3.6 Bài 6 – Thuật toán sắp xếp với visualization

**Yêu cầu:** Implement các thuật toán sắp xếp và hiển thị quá trình:
**Gợi ý cách làm:**

- Viết hàm bubble sort, selection sort, insertion sort.
- In ra mảng sau mỗi bước sắp xếp.
- So sánh số bước, thời gian thực hiện.

## 3.7 Bài 7 – Ứng dụng for/while: In mẫu ký tự (Pattern Printing)

**Yêu cầu:** Viết chương trình ứng dụng vòng lặp for, while để in ra các mẫu ký tự hình học trên màn hình console.

- Mỗi mẫu phải cho phép nhập tham số (số dòng, ký tự,...), kết quả in đúng hình và canh lề hợp lý.
- Cần tổ chức code thành các hàm riêng cho từng mẫu.
- Tạo menu để chọn mẫu, nhập tham số, in kết quả.

Chọn ít nhất 3 kiểu mẫu sau:

1. Hình vuông, hình chữ nhật
2. Hình tam giác vuông, tam giác cân
3. Hình kim tự tháp, tam giác ngược
4. Hình zíc zắc, hình thoi

**Gợi ý cách làm:**

- Nhận input từ người dùng về loại hình cần in và các tham số.
- Dùng nested loop (lồng nhau) để in từng dòng/chiều ngang.
- Kết hợp xử lý điều kiện if...else để tạo hình mong muốn.

```
Chọn một kiểu hình để in:
1. Hình vuông
2. Tam giác vuông
3. Tam giác cân
4. Kim tự tháp
5. Tam giác ngược
0. Thoát

Nhập lựa chọn: 4
Nhập số dòng (n): 7
Nhập ký tự in: *
      *
     ***
    *****
   *******
  *********
 ***********
*************
```

Hình 4: Kết quả minh họa

**Nộp bài:** Chạy code bài tập trên Colab, thực hiện lại demo và làm các bài tập về nhà theo yêu cầu, sau đó lưu dạng pdf. Đặt tên MSSV_HoVaTen_Lab3.pdf và nộp theo deadline trên Courses.