Python Programming - Cấu Trúc Điều Khiển

> **Mục tiêu học tập**: Làm chủ luồng điều khiển chương trình thông qua câu lệnh điều kiện và vòng lặp, xây dựng logic phức tạp với nested structures.

1. Câu Lệnh Điều Kiện (Conditional Statements)

1.1. Câu Lệnh `if`

🤔 Đặt vấn đề

Trong thực tế, chúng ta thường phải đưa ra quyết định dựa trên điều kiện:

- "**Nếu** điểm >= 5 **thì** đỗ, **ngược lại** thì trượt"
- "**Nếu** tuổi >= 18 **thì** được lái xe"
- "**Nếu** mưa **thì** mang ô, **ngược lại** thì không cần"

Làm sao để chương trình có thể "suy nghĩ" và ra quyết định như vậy?

💡 Giải quyết: Câu lệnh `if`

Cú pháp cơ bản:

```
if điều_kiện:
    # Khối lệnh được thực thi nếu điều kiện đúng
    câu_lệnh_1
    câu_lệnh_2
```

⚠️ Lưu ý quan trọng:

- **Dấu hai chấm `:` sau điều kiện** là BẮT BUỘC
- **Indentation (thụt đầu dòng)**: Python dùng 4 spaces để phân biệt khối lệnh
- Điều kiện phải trả về `True` hoặc `False` (hoặc truthy/falsy value)

Ví dụ đơn giản:

```
age = 20

if age >= 18:
    print("Bạn đã trưởng thành")
    print("Có thể lái xe")
```

Output:

```
Bạn đã trưởng thành
Có thể lái xe
```

Ví dụ thực tế hơn

```
score = int(input("Nhập điểm của bạn: "))

if score >= 50:
    print("🎉 Chúc mừng! Bạn đã đỗ!")
    print(f"Điểm của bạn: {score}")
```

1.2. Câu Lệnh `if-else`

🤔 Đặt vấn đề

Câu lệnh `if` đơn giản chỉ xử lý khi điều kiện đúng. Nhưng nếu điều kiện sai thì sao? Ta cần một hành động thay thế.

💡 Giải quyết: Thêm `else`

Cú pháp:

```
if điều_kiện:
    # Thực thi nếu điều kiện True
    khối_lệnh_1
else:
    # Thực thi nếu điều kiện False
    khối_lệnh_2
```

Ví dụ:

```
score = int(input("Nhập điểm: "))

if score >= 50:
    print("☑ Đỗ")
else:
    print("❌ Trượt")
```

Ví dụ phức tạp hơn:

```
age = int(input("Nhập tuổi: "))
```

```
if age >= 18:
    print("Bạn đã trưởng thành")
    print("Có thể:")
    print("- Lái xe")
    print("- Bỏ phiếu")
    print("- Ký hợp đồng")
else:
    print("Bạn chưa trưởng thành")
    print(f"Còn {18 - age} năm nữa")
```

1.3. Câu Lệnh `if-elif-else`

🤔 Đặt vấn đề

Nhiều trường hợp có **nhiều hơn 2 khả năng**. Ví dụ: xếp loại học lực có 4 mức (Giỏi, Khá, Trung bình, Yếu).

💡 Giải quyết: Dùng `elif` (else if)

Cú pháp:

```
if điều_kiện_1:
    khối_lệnh_1
elif điều_kiện_2:
    khối_lệnh_2
elif điều_kiện_3:
    khối_lệnh_3
else:
    khối_lệnh_cuối_cùng
```

⚠️ Lưu ý:

- `elif` là viết tắt của "else if"
- Có thể có **nhiều** `elif` (không giới hạn)
- `else` là tùy chọn (có thể không cần)
- Python kiểm tra **từ trên xuống dưới**, dừng lại ngay khi gặp điều kiện đầu tiên đúng

Ví dụ: Xếp loại học lực

```
score = float(input("Nhập điểm trung bình: "))

if score >= 9:
    print("🏆 Học lực: Xuất sắc")
elif score >= 8:
    print("⭐ Học lực: Giỏi")
elif score >= 6.5:
    print("👍 Học lực: Khá")
elif score >= 5:
```

```
    print("📝 Học lực: Trung bình")
else:
    print("📉 Học lực: Yếu")
```

Ví dụ: Tính phí ship dựa trên khoảng cách

```
distance = float(input("Nhập khoảng cách (km): "))

if distance <= 5:
    shipping_fee = 15000
    print(f"Phí ship: {shipping_fee:,} VND (dưới 5km)")
elif distance <= 10:
    shipping_fee = 25000
    print(f"Phí ship: {shipping_fee:,} VND (5-10km)")
elif distance <= 20:
    shipping_fee = 40000
    print(f"Phí ship: {shipping_fee:,} VND (10-20km)")
else:
    shipping_fee = 60000
    print(f"Phí ship: {shipping_fee:,} VND (trên 20km)")
```

1.4. Nested If (If lồng nhau)

🤯 Đặt vấn đề

Đôi khi ta cần kiểm tra **nhiều tầng điều kiện**. Ví dụ: Để được xét học bổng, sinh viên phải:

1. Điểm >= 8.0 VÀ
2. Không có môn nào dưới 6.5 VÀ
3. Tham gia ít nhất 80% hoạt động

💡 Giải quyết: If trong if

Ví dụ:

```
score = float(input("Điểm TB: "))

if score >= 8.0:
    print("✅ Đạt điểm TB")

    min_subject_score = float(input("Điểm môn thấp nhất: "))

    if min_subject_score >= 6.5:
        print("✅ Không có môn nào dưới 6.5")

        attendance = float(input("Tỷ lệ tham gia (%): "))

        if attendance >= 80:
```

```
            print("☑ Đạt tỷ lệ tham gia")
            print("🎓 ĐƯỢC XÉT HỌC BỔNG!")
        else:
            print("❌ Chưa đủ tỷ lệ tham gia")
    else:
        print("❌ Có môn dưới 6.5")
else:
    print("❌ Chưa đạt điểm TB")
```

⚠ **Lưu ý:** Có thể dùng toán tử `and` để viết gọn hơn:

```
score = float(input("Điểm TB: "))
min_subject = float(input("Điểm môn thấp nhất: "))
attendance = float(input("Tỷ lệ tham gia (%): "))

if score >= 8.0 and min_subject >= 6.5 and attendance >= 80:
    print("🎓 ĐƯỢC XÉT HỌC BỔNG!")
else:
    print("❌ Chưa đủ điều kiện")
```

1.5. Ternary Operator (Toán tử 3 ngôi)

🤔 Đặt vấn đề

Đôi khi ta chỉ cần gán giá trị dựa trên điều kiện đơn giản. Viết `if-else` dài dòng.

💡 Giải quyết: Ternary operator

Cú pháp:

```
giá_trị_nếu_đúng if điều_kiện else giá_trị_nếu_sai
```

So sánh:

```
# Cách thông thường
age = 20
if age >= 18:
    status = "Trưởng thành"
else:
    status = "Vị thành niên"

# Cách dùng ternary (1 dòng)
status = "Trưởng thành" if age >= 18 else "Vị thành niên"

print(status)
```

Ví dụ thực tế:

```
score = 75
result = "Đỗ" if score >= 50 else "Trượt"
print(result)  # Đỗ

# Tìm max của 2 số
a, b = 10, 20
max_value = a if a > b else b
print(f"Max: {max_value}")  # Max: 20
```

⚠ **Chú ý:** Chỉ dùng cho trường hợp đơn giản. Nếu phức tạp thì dùng `if-else` thông thường.

---

2. Vòng Lặp `for`

2.1. Khái Niệm Vòng Lặp

🤔 Đặt vấn đề

Giả sử bạn muốn:

- In ra "Hello" 10 lần
- Tính tổng từ 1 đến 100
- Duyệt qua danh sách học sinh để tính điểm trung bình

Copy-paste code 10 lần, 100 lần? **Không hiệu quả!**

💡 Giải quyết: Vòng lặp

Vòng lặp giúp thực thi một đoạn code nhiều lần mà không cần lặp lại code.

2.2. Vòng Lặp `for`

Cú pháp cơ bản

```
for biến in iterable:
    # Khối lệnh được lặp lại
    câu_lệnh
```

- `biến`: biến tạm để lưu giá trị hiện tại trong mỗi lần lặp
- `iterable`: có thể là `range()`, `list`, `string`, `tuple`, v.v.

2.3. Dùng `range()` với `for`

`range()` với 1 tham số

```
range(stop)  # Từ 0 đến stop-1
```

Ví dụ:

```
# In số từ 0 đến 4
for i in range(5):
    print(i)

# Output:
# 0
# 1
# 2
# 3
# 4
```

`range()` với 2 tham số

```
range(start, stop)  # Từ start đến stop-1
```

Ví dụ:

```
# In số từ 1 đến 10
for i in range(1, 11):
    print(i, end=" ")

# Output: 1 2 3 4 5 6 7 8 9 10
```

`range()` với 3 tham số

```
range(start, stop, step)  # Bước nhảy là step
```

Ví dụ:

```
# In số chẵn từ 0 đến 10
for i in range(0, 11, 2):
    print(i, end=" ")

# Output: 0 2 4 6 8 10
```

```
# Đếm ngược
for i in range(10, 0, -1):
    print(i, end=" ")

# Output: 10 9 8 7 6 5 4 3 2 1
```

2.4. Ứng Dụng Thực Tế Với `for`

Ví dụ 1: Tính tổng từ 1 đến n

```
n = int(input("Nhập n: "))
total = 0

for i in range(1, n + 1):
    total += i

print(f"Tổng từ 1 đến {n} = {total}")

# Với n = 10: Output = 55
```

Ví dụ 2: In bảng cửu chương

```
n = int(input("Nhập bảng cửu chương: "))

for i in range(1, 11):
    print(f"{n} x {i} = {n * i}")

# Với n = 7:
# 7 x 1 = 7
# 7 x 2 = 14
# ...
# 7 x 10 = 70
```

Ví dụ 3: Kiểm tra số nguyên tố

```
n = int(input("Nhập số cần kiểm tra: "))

if n < 2:
    print(f"{n} không phải số nguyên tố")
else:
    is_prime = True

    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
```

```
            is_prime = False
            break

    if is_prime:
        print(f"{n} là số nguyên tố")
    else:
        print(f"{n} không phải số nguyên tố")
```

Ví dụ 4: Tìm số lớn nhất trong danh sách

```
numbers = [45, 23, 89, 12, 67, 34]
max_num = numbers[0]

for num in numbers:
    if num > max_num:
        max_num = num

print(f"Số lớn nhất: {max_num}")  # 89
```

2.5. Duyệt Qua Các Cấu Trúc Dữ Liệu

Duyệt qua String

```
name = "Minh Đạo"

for char in name:
    print(char, end="-")

# Output: M-i-n-h- -Đ-ạ-o-
```

Duyệt qua List

```
fruits = ["apple", "banana", "orange"]

for fruit in fruits:
    print(f"Tôi thích {fruit}")

# Output:
# Tôi thích apple
# Tôi thích banana
# Tôi thích orange
```

Duyệt với `enumerate()` (có index)

```
students = ["An", "Bình", "Chi"]

for index, student in enumerate(students):
    print(f"{index + 1}. {student}")

# Output:
# 1. An
# 2. Bình
# 3. Chi
```

Duyệt qua Dictionary

```
scores = {
    "Toán": 8.5,
    "Lý": 7.0,
    "Hóa": 9.0
}

# Duyệt qua keys
for subject in scores:
    print(subject)

# Duyệt qua values
for score in scores.values():
    print(score)

# Duyệt qua cả key và value
for subject, score in scores.items():
    print(f"{subject}: {score}")

# Output:
# Toán: 8.5
# Lý: 7.0
# Hóa: 9.0
```

---

3. Vòng Lặp `while`

3.1. Khái Niệm

🤔 Đặt vấn đề

Vòng lặp `for` phù hợp khi biết trước số lần lặp. Nhưng nếu:

- Lặp đến khi người dùng nhập "quit"
- Lặp đến khi tìm được kết quả
- Lặp đến khi điều kiện nào đó không còn đúng

Làm sao biết trước bao nhiêu lần lặp?

Giải quyết: Vòng lặp `while`

Cú pháp:

```
while điều_kiện:
    # Khối lệnh lặp lại KHI điều kiện còn True
    câu_lệnh
```

⚠ **Cẩn thận:** Phải đảm bảo điều kiện sẽ trở thành `False`, nếu không sẽ **vòng lặp vô tận**.

3.2. Ví Dụ Cơ Bản

Đếm từ 1 đến 5

```
count = 1

while count <= 5:
    print(f"Lần {count}")
    count += 1  # QUAN TRỌNG: Thay đổi biến để thoát vòng lặp

print("Xong!")

# Output:
# Lần 1
# Lần 2
# Lần 3
# Lần 4
# Lần 5
# Xong!
```

Tính tổng đến khi đủ điều kiện

```
total = 0
number = 1

while total < 100:
    total += number
    number += 1

print(f"Tổng đầu tiên >= 100: {total}")
print(f"Cần cộng đến số: {number - 1}")

# Output:
```

```
# Tổng đầu tiên >= 100: 105
# Cần cộng đến số: 14
```

3.3. Ứng Dụng Thực Tế

Ví dụ 1: Menu lựa chọn

```
while True:
    print("\n=== MENU ===")
    print("1. Tính tổng")
    print("2. Tính hiệu")
    print("3. Tính tích")
    print("4. Thoát")

    choice = input("Chọn (1-4): ")

    if choice == "4":
        print("Tạm biệt!")
        break
    elif choice in ["1", "2", "3"]:
        a = float(input("Số thứ nhất: "))
        b = float(input("Số thứ hai: "))

        if choice == "1":
            print(f"Kết quả: {a + b}")
        elif choice == "2":
            print(f"Kết quả: {a - b}")
        elif choice == "3":
            print(f"Kết quả: {a * b}")
    else:
        print("Lựa chọn không hợp lệ!")
```

Ví dụ 2: Xác thực mật khẩu

```
password = "python123"
attempts = 3

while attempts > 0:
    user_input = input(f"Nhập mật khẩu ({attempts} lần thử còn lại): ")

    if user_input == password:
        print("✅ Đăng nhập thành công!")
        break
    else:
        attempts -= 1
        if attempts > 0:
            print(f"❌ Sai mật khẩu. Còn {attempts} lần thử.")
```

```
        else:
            print("❌ Hết lượt thử. Tài khoản bị khóa!")
```

Ví dụ 3: Đọc input cho đến khi hợp lệ

```
while True:
    age_input = input("Nhập tuổi (1-120): ")

    if age_input.isdigit():
        age = int(age_input)
        if 1 <= age <= 120:
            print(f"Tuổi của bạn: {age}")
            break
        else:
            print("Tuổi phải từ 1 đến 120!")
    else:
        print("Vui lòng nhập số!")
```

3.4. So Sánh `for` vs `while`

Tiêu chí	`for`	`while`
Khi nào dùng	Biết trước số lần lặp	Không biết trước số lần lặp
Ví dụ	Duyệt list, range	Lặp đến khi user nhập "quit"
Rủi ro	Ít rủi ro	Dễ gây vòng lặp vô tận
Cú pháp	Ngắn gọn hơn	Linh hoạt hơn

Ví dụ tương đương:

```
# for
for i in range(5):
    print(i)

# while (tương đương)
i = 0
while i < 5:
    print(i)
    i += 1
```

---

4. `break` và `continue`

4.1. Lệnh `break`

🤔 Đặt vấn đề

Đôi khi ta cần **thoát khỏi vòng lặp ngay lập tức**, không cần chờ điều kiện kết thúc tự nhiên.

💡 `break` - Thoát vòng lặp

Ví dụ 1: Tìm số đầu tiên chia hết cho 7

```
for i in range(1, 101):
    if i % 7 == 0:
        print(f"Số đầu tiên chia hết cho 7: {i}")
        break  # Thoát vòng lặp ngay

# Output: Số đầu tiên chia hết cho 7: 7
```

Ví dụ 2: Tìm kiếm trong list

```
students = ["An", "Bình", "Chi", "Dũng"]
search_name = "Chi"
found = False

for student in students:
    if student == search_name:
        print(f"Tìm thấy {search_name}!")
        found = True
        break

if not found:
    print(f"Không tìm thấy {search_name}")
```

Ví dụ 3: Break trong while

```
count = 0

while True:  # Vòng lặp vô tận
    print(f"Lần {count + 1}")
    count += 1

    if count >= 5:
        break  # Thoát khi đủ 5 lần

print("Đã thoát vòng lặp")
```

4.2. Lệnh `continue`

🤔 Đặt vấn đề

Đôi khi ta muốn **bỏ qua lần lặp hiện tại** và tiếp tục với lần lặp kế tiếp, không phải thoát hẳn vòng lặp.

💡 `continue` - Bỏ qua lần lặp hiện tại

Ví dụ 1: In số lẻ từ 1 đến 10

```
for i in range(1, 11):
    if i % 2 == 0:
        continue  # Bỏ qua số chẵn
    print(i, end=" ")

# Output: 1 3 5 7 9
```

Ví dụ 2: Bỏ qua số âm

```
numbers = [10, -5, 20, -3, 15, -8]

for num in numbers:
    if num < 0:
        continue  # Bỏ qua số âm
    print(num, end=" ")

# Output: 10 20 15
```

Ví dụ 3: Xử lý danh sách với điều kiện

```
students = ["An", "Bình", "", "Chi", "Dũng", ""]

for i, student in enumerate(students, 1):
    if student == "":
        continue  # Bỏ qua tên rỗng
    print(f"{i}. {student}")

# Output:
# 1. An
# 2. Bình
# 4. Chi
# 5. Dũng
```

4.3. So Sánh `break` vs `continue`

Lệnh	Chức năng	Khi nào dùng

Lệnh	Chức năng	Khi nào dùng
`break`	**Thoát** khỏi vòng lặp hoàn toàn	Đã tìm thấy kết quả, không cần tiếp tục
`continue`	**Bỏ qua** lần lặp hiện tại, tiếp tục lần sau	Lọc bỏ các giá trị không mong muốn

Ví dụ minh họa:

```
# break: Dừng khi gặp số 5
for i in range(10):
    if i == 5:
        break
    print(i, end=" ")
# Output: 0 1 2 3 4

print("\n---")

# continue: Bỏ qua số 5
for i in range(10):
    if i == 5:
        continue
    print(i, end=" ")
# Output: 0 1 2 3 4 6 7 8 9
```

---

5. Vòng Lặp Lồng Nhau (Nested Loops)

5.1. Khái Niệm

🤯 Đặt vấn đề

Đôi khi ta cần **vòng lặp trong vòng lặp**. Ví dụ:

- In bảng cửu chương từ 1 đến 10 (mỗi bảng có 10 phép tính)
- Duyệt qua ma trận 2D (mảng 2 chiều)
- Tạo các cặp tổ hợp

💡 Nested Loops

Cú pháp:

```
for i in range(n):          # Vòng lặp ngoài
    for j in range(m):      # Vòng lặp trong
        # Khối lệnh
```

⚠ **Lưu ý:** Vòng lặp trong sẽ chạy **hoàn toàn** cho mỗi lần lặp của vòng lặp ngoài.

5.2. Ví Dụ Cơ Bản

In hình chữ nhật sao

```
rows = 4
cols = 6

for i in range(rows):
    for j in range(cols):
        print("*", end=" ")
    print()  # Xuống dòng sau mỗi hàng

# Output:
# * * * * * *
# * * * * * *
# * * * * * *
# * * * * * *
```

In tam giác sao

```
n = 5

for i in range(1, n + 1):
    for j in range(i):
        print("*", end=" ")
    print()

# Output:
# *
# * *
# * * *
# * * * *
# * * * * *
```

5.3. Ứng Dụng Thực Tế

Ví dụ 1: In bảng cửu chương đầy đủ

```
print("=== BẢNG CỬU CHƯƠNG ===\n")

for i in range(1, 11):  # Bảng từ 1 đến 10
    print(f"--- Bảng {i} ---")
    for j in range(1, 11):  # Mỗi bảng có 10 phép tính
        print(f"{i} x {j} = {i * j}")
    print()
```

Ví dụ 2: Duyệt ma trận 2D

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Duyệt qua từng phần tử
for i in range(len(matrix)):        # Duyệt hàng
    for j in range(len(matrix[i])): # Duyệt cột
        print(f"matrix[{i}][{j}] = {matrix[i][j]}")

# Output:
# matrix[0][0] = 1
# matrix[0][1] = 2
# ...
# matrix[2][2] = 9
```

Ví dụ 3: Tìm tất cả các cặp số có tổng = target

```
numbers = [2, 7, 11, 15, 3, 6]
target = 9

print(f"Các cặp có tổng = {target}:")

for i in range(len(numbers)):
    for j in range(i + 1, len(numbers)):  # Tránh lặp lại cặp
        if numbers[i] + numbers[j] == target:
            print(f"{numbers[i]} + {numbers[j]} = {target}")

# Output:
# 2 + 7 = 9
# 3 + 6 = 9
```

Ví dụ 4: In tam giác số

```
n = 5

for i in range(1, n + 1):
    for j in range(1, i + 1):
        print(j, end=" ")
    print()

# Output:
# 1
# 1 2
```

```
# 1 2 3
# 1 2 3 4
# 1 2 3 4 5
```

5.4. `break` và `continue` Trong Nested Loops

`break` chỉ thoát vòng lặp gần nhất

```
for i in range(3):
    print(f"Vòng ngoài: {i}")
    for j in range(3):
        if j == 1:
            break  # Chỉ thoát vòng trong
        print(f"  Vòng trong: {j}")

# Output:
# Vòng ngoài: 0
#   Vòng trong: 0
# Vòng ngoài: 1
#   Vòng trong: 0
# Vòng ngoài: 2
#   Vòng trong: 0
```

Thoát cả hai vòng lặp với flag

```
found = False

for i in range(5):
    for j in range(5):
        if i * j == 12:
            print(f"Tìm thấy: {i} x {j} = 12")
            found = True
            break
    if found:
        break
```

`continue` trong nested loop

```
for i in range(3):
    for j in range(3):
        if j == 1:
            continue  # Bỏ qua j=1, tiếp tục j=2
        print(f"i={i}, j={j}")

# Output:
```

```
# i=0, j=0
# i=0, j=2
# i=1, j=0
# i=1, j=2
# i=2, j=0
# i=2, j=2
```

---

6. Bài Tập Thực Hành

Bài 1: Kiểm tra số chẵn/lẻ trong khoảng

```
"""
Input: Hai số a, b (a < b)
Output: In ra các số chẵn trong khoảng [a, b]
"""

a = int(input("Nhập a: "))
b = int(input("Nhập b: "))

print(f"Các số chẵn từ {a} đến {b}:")
for i in range(a, b + 1):
    if i % 2 == 0:
        print(i, end=" ")
```

Bài 2: Tính giai thừa

```
"""
Input: Số nguyên dương n
Output: n! (n giai thừa)
Ví dụ: 5! = 5 × 4 × 3 × 2 × 1 = 120
"""

n = int(input("Nhập n: "))
factorial = 1

for i in range(1, n + 1):
    factorial *= i

print(f"{n}! = {factorial}")
```

Bài 3: Tìm ước số

```
"""
Input: Số nguyên dương n
Output: Tất cả các ước số của n
"""

n = int(input("Nhập n: "))
print(f"Các ước số của {n}:")

for i in range(1, n + 1):
    if n % i == 0:
        print(i, end=" ")
```

Bài 4: Tính tổng các chữ số

```
"""
Input: Số nguyên dương n
Output: Tổng các chữ số
Ví dụ: 12345 → 1+2+3+4+5 = 15
"""

n = int(input("Nhập số: "))
total = 0

while n > 0:
    digit = n % 10  # Lấy chữ số cuối
    total += digit
    n //= 10        # Bỏ chữ số cuối

print(f"Tổng các chữ số: {total}")
```

Bài 5: Đảo ngược số

```
"""
Input: Số nguyên dương n
Output: Số đảo ngược
Ví dụ: 12345 → 54321
"""

n = int(input("Nhập số: "))
reversed_num = 0

while n > 0:
    digit = n % 10
    reversed_num = reversed_num * 10 + digit
    n //= 10
```

```
print(f"Số đảo ngược: {reversed_num}")
```

Bài 6: In hình kim cương

```
"""
Input: n (chiều cao)
Output: Hình kim cương bằng dấu *
"""

n = int(input("Nhập chiều cao: "))

# Nửa trên (bao gồm giữa)
for i in range(n):
    # In khoảng trắng
    for j in range(n - i - 1):
        print(" ", end="")
    # In dấu sao
    for j in range(2 * i + 1):
        print("*", end="")
    print()

# Nửa dưới
for i in range(n - 2, -1, -1):
    for j in range(n - i - 1):
        print(" ", end="")
    for j in range(2 * i + 1):
        print("*", end="")
    print()

# Với n = 5:
#     *
#    ***
#   *****
#  *******
# *********
#  *******
#   *****
#    ***
#     *
```

Bài 7: Đếm số nguyên tố trong khoảng

```
"""
Input: Hai số a, b
Output: Số lượng số nguyên tố trong [a, b]
"""
```

```python
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True

a = int(input("Nhập a: "))
b = int(input("Nhập b: "))

count = 0
print(f"Các số nguyên tố từ {a} đến {b}:")

for num in range(a, b + 1):
    if is_prime(num):
        print(num, end=" ")
        count += 1

print(f"\nTổng cộng: {count} số nguyên tố")
```

Bài 8: Tìm số hoàn hảo

```python
"""
Số hoàn hảo: tổng các ước (không kể chính nó) = chính nó
Ví dụ: 6 = 1 + 2 + 3
Input: n
Output: Các số hoàn hảo <= n
"""

n = int(input("Nhập n: "))
print(f"Các số hoàn hảo <= {n}:")

for num in range(1, n + 1):
    divisor_sum = 0

    # Tính tổng các ước (không kể chính nó)
    for i in range(1, num):
        if num % i == 0:
            divisor_sum += i

    if divisor_sum == num:
        print(num, end=" ")

# Với n = 1000: Output = 6 28 496
```

---

7. Pattern Printing (In Hình)

Pattern 1: Tam giác vuông

```
n = 5
for i in range(1, n + 1):
    for j in range(i):
        print("*", end=" ")
    print()

# *
# * *
# * * *
# * * * *
# * * * * *
```

Pattern 2: Tam giác vuông ngược

```
n = 5
for i in range(n, 0, -1):
    for j in range(i):
        print("*", end=" ")
    print()

# * * * * *
# * * * *
# * * *
# * *
# *
```

Pattern 3: Tam giác cân

```
n = 5
for i in range(1, n + 1):
    # Khoảng trắng
    for j in range(n - i):
        print(" ", end=" ")
    # Dấu sao
    for j in range(i):
        print("* ", end="")
    print()

#         *
#       * *
#     * * *
#   * * * *
# * * * * *
```

Pattern 4: Số tăng dần

```
n = 5
num = 1
for i in range(1, n + 1):
    for j in range(i):
        print(num, end=" ")
        num += 1
    print()

# 1
# 2 3
# 4 5 6
# 7 8 9 10
# 11 12 13 14 15
```

---

8. Tổng Kết và Checklist

☑ Kiến thức cần nắm vững

Câu lệnh điều kiện:

- ☐ `if` đơn giản
- ☐ `if-else` (2 nhánh)
- ☐ `if-elif-else` (nhiều nhánh)
- ☐ Nested if (if lồng nhau)
- ☐ Ternary operator (toán tử 3 ngôi)

Vòng lặp `for`:

- ☐ Cú pháp cơ bản với `range()`
- ☐ `range(start, stop, step)`
- ☐ Duyệt qua list, string, dictionary
- ☐ `enumerate()` để có index

Vòng lặp `while`:

- ☐ Cú pháp và cách sử dụng
- ☐ Tránh vòng lặp vô tận
- ☐ So sánh với `for`

`break` và `continue`:

- ☐ `break`: thoát vòng lặp
- ☐ `continue`: bỏ qua lần lặp hiện tại
- ☐ Sử dụng trong nested loops

Nested Loops:

- ☐ Vòng lặp lồng nhau
- ☐ In các pattern (hình)
- ☐ Duyệt ma trận 2D

🎯 Lưu ý quan trọng

1. **Indentation (thụt đầu dòng)**: Python dùng 4 spaces để phân biệt khối lệnh
2. **Dấu hai chấm `:`**: Bắt buộc sau `if`, `for`, `while`, `elif`, `else`
3. **Tránh vòng lặp vô tận**: Luôn đảm bảo điều kiện `while` sẽ trở thành `False`
4. `break` **vs** `continue`: break = thoát hẳn, continue = bỏ qua lần này
5. **Nested loops**: Độ phức tạp $O(n^2)$, cẩn thận với n lớn

📚 Bước tiếp theo

Sau khi nắm vững phần này, bạn sẵn sàng học:

- Functions (hàm)
- List comprehensions
- Exception handling
- Cấu trúc dữ liệu nâng cao

---

9. Câu Hỏi Thường Gặp (FAQ)

Q1: Sự khác biệt chính giữa `for` và `while`?

A:

- `for`: Dùng khi **biết trước số lần lặp** (duyệt list, range)
- `while`: Dùng khi **không biết trước** (lặp đến khi điều kiện thỏa mãn)

```
# for: Biết trước lặp 10 lần
for i in range(10):
    print(i)

# while: Không biết trước bao nhiêu lần
while user_input != "quit":
    user_input = input("Nhập (quit để thoát): ")
```

Q2: `range(5)` **và** `range(1, 6)` **khác nhau thế nào?**

A:

- `range(5)`: từ 0 đến 4 (5 số)
- `range(1, 6)`: từ 1 đến 5 (5 số)

```
list(range(5))      # [0, 1, 2, 3, 4]
list(range(1, 6))   # [1, 2, 3, 4, 5]
```

Q3: Làm sao để thoát khỏi cả 2 vòng lặp lồng nhau?

A: Dùng biến flag hoặc đưa vào function và dùng `return`.

```
# Cách 1: Flag
found = False
for i in range(10):
    for j in range(10):
        if i * j == 50:
            found = True
            break
    if found:
        break

# Cách 2: Function
def find_product():
    for i in range(10):
        for j in range(10):
            if i * j == 50:
                return i, j
    return None, None
```

Q4: Khi nào dùng `break` và khi nào dùng `continue`?

A:

- `break`: Đã tìm thấy kết quả, không cần lặp nữa
- `continue`: Bỏ qua một số trường hợp, vẫn tiếp tục lặp

```
# break: Tìm số đầu tiên > 50
for num in numbers:
    if num > 50:
        print(f"Tìm thấy: {num}")
        break

# continue: Bỏ qua số âm
for num in numbers:
    if num < 0:
        continue
    print(num)
```

Q5: Tại sao code của tôi bị "IndentationError"?

A: Python **rất nghiêm ngặt** về indentation. Phải dùng:

- **4 spaces** (hoặc 1 tab)
- **Nhất quán** trong toàn bộ file

```
# ❌ SAI - Thiếu indent
if x > 0:
print(x)

# ❌ SAI - Indent không đều
if x > 0:
    print("Dương")
  print("Giá trị:", x)  # Chỉ 2 spaces

# ☑ ĐÚNG
if x > 0:
    print("Dương")
    print("Giá trị:", x)
```

---

10. Thử Thách Cuối Khóa

Challenge 1: Trò chơi đoán số

Viết chương trình:

- Máy chọn ngẫu nhiên số từ 1-100
- Người chơi đoán, máy báo "cao hơn" hoặc "thấp hơn"
- Giới hạn 7 lần đoán

```
"""
Gợi ý:
- import random
- random.randint(1, 100)
- Dùng while loop và counter
"""
```

Challenge 2: Kiểm tra palindrome

Viết chương trình kiểm tra chuỗi có phải palindrome không (đọc xuôi ngược giống nhau).

```
"""
Ví dụ:
- "madam" → True
- "racecar" → True
- "hello" → False

Gợi ý: So sánh chuỗi với chuỗi đảo ngược
"""
```

Challenge 3: In hình cây thông Noel

```
"""
Input: Chiều cao n
Output: Hình cây thông

Ví dụ với n = 5:
    *
   ***
  *****
 *******
*********
    *
    *
"""
```

Challenge 4: Tìm số Fibonacci thứ n

```
"""
Dãy Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...
Công thức: F(n) = F(n-1) + F(n-2)

Input: n
Output: Số Fibonacci thứ n
"""
```
