Tuần 1 - Tổng hợp kiến thức Buổi học số 3

Time-Series Team

Ngày 8 tháng 6 năm 2025

Buổi học số 4 (Thứ 6, 06/06/2025) bao gồm bốn nội dung chính:

- ***Phần 1: Các lỗi thường gặp trong Python***
- ***Phần 2: Cấu trúc và Cách thức hoạt động vòng lặp For***
- ***Phần 3: Cấu trúc và Cách thức hoạt động vòng lặp While***
- ***Phần 4: Toán mở rộng và Ứng dụng vòng lặp trong công thức Toán học***

Phần I: Các lỗi thường gặp trong Python

1 SyntaxError (Lỗi cú pháp)

Nguyên nhân: Viết sai cú pháp, thiếu dấu hoặc sai cấu trúc câu lệnh. **Ví dụ:**

```
print("Hello world"
```

Sửa:

```
print("Hello world")
```

1.1 IndentationError (Lỗi thụt đầu dòng)

Nguyên nhân: Thụt đầu dòng không đúng quy tắc (Python yêu cầu thụt đầu dòng chính xác). **Ví dụ:**

```
def hello():
print("Hi")
```

Sửa:

```
def hello():
    print("Hi")
```

2 NameError (Lỗi tên biến không định nghĩa)

Nguyên nhân: Sử dụng biến hoặc hàm chưa được khai báo. **Ví dụ:**

```
print(x)
```

Sửa:

```
x = 10
print(x)
```

3 TypeError (Lỗi kiểu dữ liệu không phù hợp)

Nguyên nhân: Thực hiện phép toán hoặc gọi hàm với kiểu dữ liệu sai. **Ví dụ:**

```
"hello" + 5
```

Sửa:

```
"hello" + str(5)
```

4 ValueError (Lỗi giá trị không phù hợp)

Nguyên nhân: Giá trị truyền vào hàm hợp lệ về kiểu nhưng không hợp lý. **Ví dụ:**

```
int("abc")
```

Sửa:

```
int("123")
```

5 IndexError (Lỗi truy cập chỉ số ngoài phạm vi)

Nguyên nhân: Truy cập phần tử vượt quá chỉ số của list hoặc chuỗi. **Ví dụ:**

```
lst = [1, 2, 3]
print(lst[3])
```

Sửa:

```
print(lst[2])
```

6 KeyError (Lỗi truy cập khóa không tồn tại trong dict)

Nguyên nhân: Truy cập key không có trong dictionary. **Ví dụ:**

```
d = {"a": 1, "b": 2}
print(d["c"])
```

Sửa:

```
print(d.get("c"))
```

7 AttributeError (Lỗi gọi thuộc tính hoặc phương thức không tồn tại)

Nguyên nhân: Gọi phương thức hoặc thuộc tính không có trên đối tượng. **Ví dụ:**

```
x = 5
x.append(3)
```

Sửa:

```
lst = [5]
lst.append(3)
```

8 ZeroDivisionError (Lỗi chia cho 0)

Nguyên nhân: Chia một số cho 0 trong phép toán. **Ví dụ:**

```
x = 10 / 0
```

Sửa:

```
if denominator != 0:
    x = 10 / denominator
else:
    print("Không chia cho 0")
```

9 ImportError / ModuleNotFoundError (Lỗi import module)

Nguyên nhân: Module không tồn tại hoặc chưa được cài đặt. **Ví dụ:**

```
import non_existing_module
```

Sửa: Cài module đúng hoặc sửa tên module chính xác.

Phần II: Vòng lặp For trong Python

1 Khái niệm For Loop

Vòng lặp `for` cho phép lặp qua các phần tử trong một *iterable* (chuỗi, danh sách, tuple, dictionary, hay `range`).

```
for i in range(5):
  print("hello")
```

Câu lệnh For cơ bản

Kết quả:

```
hello
hello
hello
hello
hello
```

2 Cấu trúc vòng lặp For

- **Keyword:** `for`
- **Colon:** Dấu hai chấm `:` sau `for`
- **Indentation:** Thụt lệnh là bắt buộc
- **Iterables:** `String, Tuple, List, Dictionary, range()`

3 Cách hoạt động của For Loop

- Kiểm tra xem đã dịch qua hết phần tử chưa.
- Thực thi các lệnh trong vòng `for`.
- Khi hết dữ liệu: thoát vòng lặp.

4 Ví dụ với vòng lặp For

Tính bình phương các số

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

Kết quả: `0 1 4 9 16`

Tính tổng

```
total = 0
for i in range(5):
    total = total + i
print(total)
```

Kết quả: `0 1 3 6 10`

Tính giai thừa (factorial)

```
result = 1
for i in range(1, 5):
    result = result * i
print(result)
```

Kết quả: `1 2 6 24`

5 Các hàm và từ khóa phổ biến trong vòng lặp for

5.1 Hàm range()

Hàm `range()` trong Python được dùng để sinh ra một dãy số nguyên, thường dùng trong vòng lặp `for`. Nó không tạo ra một danh sách thật sự mà trả về một đối tượng *range object* – hiệu quả hơn về bộ nhớ.

5.1.1 Cú pháp

```
range(stop)
range(start, stop)
range(start, stop, step)
```

- **start (tùy chọn)**: Giá trị bắt đầu, mặc định là `0`.
- **stop (bắt buộc)**: Giá trị kết thúc (không bao gồm `stop`).
- **step (tùy chọn)**: Bước nhảy giữa các giá trị, mặc định là `1`. Có thể là số âm.

5.1.2 Ví dụ sử dụng

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

range(stop)

Kết quả: `0 1 2 3 4`

```
for i in range(2, 6):
    print(i)
```

range(start, stop)

Kết quả: `2 3 4 5`

```
for i in range(1, 10, 2):
    print(i)
```

range(start, stop, step)

Kết quả: `1 3 5 7 9`

```
for i in range(5, 0, -1):
    print(i)
```

Dãy số giảm dần

Kết quả: `5 4 3 2 1`

5.1.3 Chuyển range thành danh sách

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

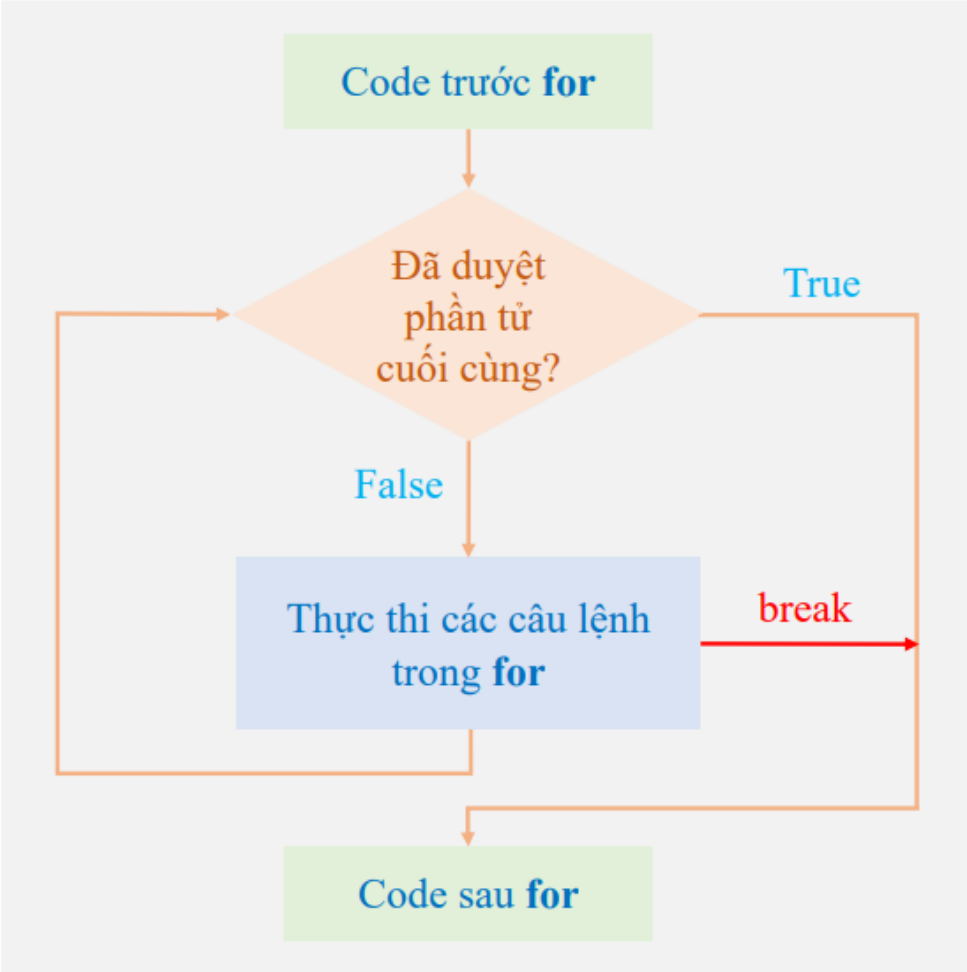
Ghi chú: Trong Python 3, `range()` sinh ra một *range object* → không chiếm nhiều bộ nhớ như list lớn.

5.2 Từ khoá break

`break` dùng để thoát vòng lặp khi đúng điều kiện.

```
for i in range(5):
    if i == 2:
    break
    print(i)
```

Kết quả: `0 1`



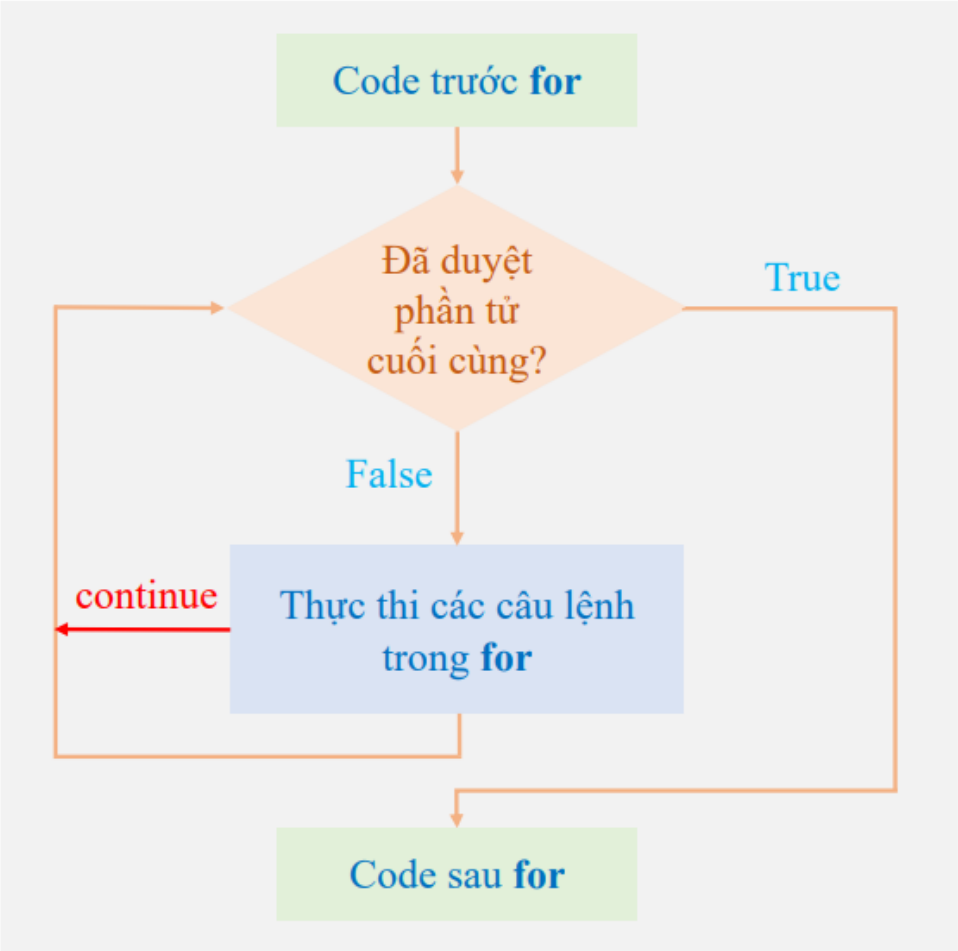
Hình 1: Mô tả hoạt động vòng lặp For với break

5.3 Từ khoá continue

`continue` bỏ qua vòng lặp hiện tại và chuyển sang lần lặp kế tiếp.

```
for i in range(5):
    if i == 2:
    continue
    print(i)
```

Kết quả: `0 1 3 4`

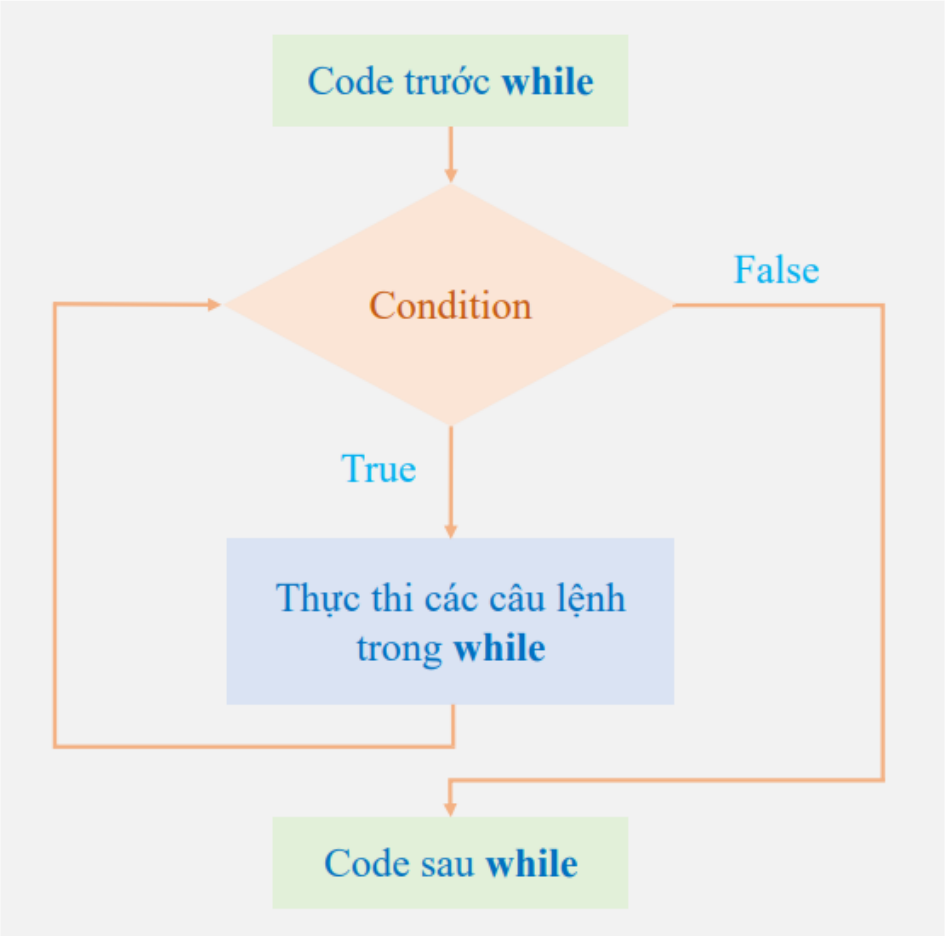


Hình 2: Mô tả hoạt động vòng lặp For với continue

Phần II: Vòng lặp While trong Python

1 Khái niệm vòng lặp While

Vòng lặp `while` thực hiện một khối lệnh lặp đi lặp lại **miễn là điều kiện còn đúng (True)**. Khi điều kiện trở thành False, vòng lặp sẽ dừng lại.



Hình 3: Cấu trúc hoạt động vòng lặp while

2 Cấu trúc cơ bản vòng lặp While

```
while condition:
    # loop body
```

Cấu trúc vòng lặp while

Thành phần:

- `keyword`: `while` là từ khóa bắt buộc.
- `condition`: Biểu thức điều kiện (boolean).
- `colon`: Dấu hai chấm `:` sau điều kiện.

- `indentation`: Các câu lệnh bên trong vòng lặp phải được thụt đầu dòng.

2.1 Ví dụ ứng dụng vòng lặp While

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

In ra các số từ 0 đến 4

Kết quả:

```
0
1
2
3
4
```

Cách hoạt động của vòng lặp:

1. Khởi tạo giá trị ban đầu (ví dụ: `i = 0`).
2. Kiểm tra điều kiện `i < 5`.
3. Nếu điều kiện đúng:
   - Thực thi khối lệnh bên trong.
   - Cập nhật giá trị `i`.
4. Quay lại bước 2 cho đến khi điều kiện sai.
5. Khi điều kiện sai → thoát vòng lặp → chạy tiếp phần code bên dưới.

3 Vòng lặp While với Từ khóa

Ví dụ: sử dụng `while True` và `break`

```
while True:
    n = int(input("Positive number: "))
    if n > 0:
        break
```

Giải thích: Vòng lặp tiếp tục mãi cho đến khi người dùng nhập số dương → dùng `break` để thoát vòng lặp.

Ví dụ: sử dụng `while True` và `continue`

```
while True:
    n = int(input("Negative number: "))
    if n > 0:
        continue
```

Giải thích: Vòng lặp tiếp tục mãi cho đến khi người dùng nhập số âm → dùng `continue` để tiếp tục vòng lặp.

Phần IV: Toán mở rộng và Ứng dụng vòng lặp trong công thức Toán học

1 Gregory-Leibniz Series

Gregory-Leibniz Series

Công thức Gregory–Leibniz thường được sử dụng để **xấp xỉ** giá trị của số π dưới dạng:

$$\pi \approx 4 \sum_{i=1}^{n} \frac{(-1)^{i+1}}{2i-1}$$

Tuy nhiên, khi số lượng phần tử n tiến đến vô hạn, công thức này thực sự trở thành một **đẳng thức chính xác**:

$$\pi = 4 \sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{2i-1}$$

Chứng minh:
Dựa vào khai triển **Maclaurin** của hàm $\arctan(x)$:

$$\arctan(x) = \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} x^{2i+1}, \quad \text{với } |x| \leq 1$$

Khi thay $x = 1$, ta được:

$$\arctan(1) = \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = \frac{\pi}{4}$$

Suy ra, nhân hai vế với 4, ta có:

$$\pi = 4 \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} \qquad (\square)$$

Sử dụng vòng lặp `for`

```python
n = 100000
pi = 0
for k in range(n):
    pi += (-1)**k / (2 * k + 1)

pi *= 4
print(f"Approximate value of pi (for): {pi}")
```

Code Listing 1: Tính gần đúng π với `for`

Sử dụng vòng lặp `while`

```python
epsilon = 1e-6
pi = 0
k = 0
term = 1  # start loop

```

```python
while abs(term) > epsilon:
    term = (-1)**k / (2 * k + 1)
    pi += term
    k += 1

pi *= 4
print(f"Approximate value of pi (while): {pi}")
print(f"The number of loops: {k}")
```

Code Listing 2: Tính gần đúng π với `while`

So sánh

- Với `for`: kiểm soát số lượng bước lặp cố định.
- Với `while`: kiểm soát độ chính xác bằng sai số `epsilon`.

2 Nilakantha Series

Dù công thức Gregory–Leibniz rất nổi tiếng vì sự đơn giản, nhưng tốc độ hội tụ của nó lại là một nhược điểm lớn. Để khắc phục điều đó, Nilakantha đã đưa ra một chuỗi khác có tốc độ hội tụ tốt hơn, bắt đầu từ số 3 và lần lượt cộng trừ các phân số có mẫu là tích 3 số liên tiếp:

$$\pi = 3 + \frac{4}{2 \cdot 3 \cdot 4} - \frac{4}{4 \cdot 5 \cdot 6} + \frac{4}{6 \cdot 7 \cdot 8} - \cdots \tag{1}$$

hay ta có:

$$\pi \approx 3 + \sum_{i=0}^{n} \frac{(-1)^{i+1} \cdot 4}{(2i+2)(2i+3)(2i+4)}$$

Nilakantha Series

Khi số lượng số hạng tiến tới vô hạn, biểu thức trên trở thành một đẳng thức chính xác cho số π:

$$\pi = 3 + \sum_{i=1}^{\infty} \frac{(-1)^{i+1} \cdot 4}{(2i+2)(2i+3)(2i+4)} \tag{*}$$

Chứng minh:

Từ công thức (*), ta cần chứng minh:

$$\sum_{i=1}^{\infty} \frac{(-1)^{i+1} \cdot 4}{(2i+2)(2i+3)(2i+4)} = \pi - 3 \tag{2}$$

Để chứng minh công thức trên, ta cần có bổ đề sau:

Công thức tích phân Beta (dạng đẳng thức)

Khi $m, n \in \mathbb{N}$, tích phân Beta có thể được biểu diễn dưới dạng:

$$\int_0^1 t^m (1-t)^n \, dt = \frac{m! \cdot n!}{(m+n+1)!} \tag{3}$$

Ta đặt:

- $m = 2i + 1$
- $n = 2$

Áp dụng công thức Beta:

$$\int_0^1 t^{2i+1}(1-t)^2 \, dt \overset{(3)}{=} \frac{(2i+1)! \cdot 2!}{(2i+1+2+1)!} = \frac{2(2i+1)!}{(2i+4)!}$$

Mặt khác, ta có:

$$(2i+2)(2i+3)(2i+4) = \frac{(2i+4)!}{(2i+1)!} \Rightarrow \frac{1}{(2i+2)(2i+3)(2i+4)} = \frac{(2i+1)!}{(2i+4)!}$$

Do đó:

$$\int_0^1 t^{2i+1}(1-t)^2 \, dt = \frac{2(2i+1)!}{(2i+4)!} \Rightarrow \frac{1}{(2i+2)(2i+3)(2i+4)} = \frac{1}{2}\int_0^1 t^{2i+1}(1-t)^2 \, dt \tag{4}$$

Mặt khác, ta có công thức khai triển Maclaurin của $\dfrac{1}{1+t^2}$:

$$\frac{1}{1+t^2} = \sum_{i=0}^{\infty} (-1)^i t^{2i}, \quad \text{với } |t| < 1$$

Nhân cả hai vế với t, ta được:

$$\frac{t}{1+t^2} = \sum_{i=0}^{\infty} (-1)^i t^{2i+1} \tag{5}$$

Từ đây ta có thể biến đổi chuỗi ban đầu:

$$\begin{aligned}
\sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+2)(2i+3)(2i+4)} &\overset{(4)}{=} \frac{1}{2}\sum_{i=0}^{\infty}(-1)^i \int_0^1 t^{2i+1}(1-t)^2 \, dt \\
&= \frac{1}{2}\int_0^1 \left(\sum_{i=0}^{\infty}(-1)^i t^{2i+1}\right)(1-t)^2 \, dt \\
&\overset{(5)}{=} \frac{1}{2}\int_0^1 \frac{t(1-t)^2}{1+t^2} \, dt = \frac{-3+\pi}{4}
\end{aligned}$$

Vậy ta có:

$$\sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+2)(2i+3)(2i+4)} = \frac{-3+\pi}{4}$$

hay công thức (2) đúng. (□)

Sử dụng vòng lặp `for`

```python
n = 100000
pi = 3.0
sign = 1

for i in range(2, 2*n, 2):
    term = 4 / (i * (i+1) * (i+2))
    pi += sign * term
    sign *= -1

print(f"Approximate value of pi (for): {pi}")
```

Code Listing 3: Tính gần đúng π bằng chuỗi Nilakantha (for)

Sử dụng vòng lặp `while`

```python
epsilon = 1e-8
pi = 3.0
i = 2
sign = 1
term = 1  # ởkhi ạto để ắbt đầu vòng ặlp

while abs(term) > epsilon:
    term = 4 / (i * (i+1) * (i+2))
    pi += sign * term
    sign *= -1
    i += 2

print(f"Approximate value of pi (while): {pi}")
print(f"The number of loops: {(i-2)//2}")
```

Code Listing 4: Tính gần đúng π bằng chuỗi Nilakantha (while)

Ghi chú

- Chuỗi hội tụ nhanh hơn chuỗi Gregory–Leibniz.
- Sử dụng sai số `epsilon` để kiểm soát độ chính xác trong `while`.

3 Quadratic Root

Chúng ta cần giải quyết bài toán **tìm căn bậc hai (square root)** của một số $a > 0$ bằng **phương pháp Newton (Newton-Raphson)**. Dưới dạng toán học, ta có thể phát biểu lại như sau:

Bài toán

Tìm nghiệm dương của phương trình:

$$x^2 = a \quad \Leftrightarrow \quad x = \sqrt{a}$$

Ta áp dụng phương pháp Newton để tìm nghiệm của hàm:

$$f(x) = x^2 - a$$

Đạo hàm:

$$f'(x) = 2x$$

Sử dụng công thức lặp Newton:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$$

Từ công thức trên, chúng ta cần chứng minh dãy x_n hội tụ về $\sqrt{a}$ khi $n \to \infty$. Hay:

$$\lim_{n\to\infty} x_n^2 = a$$

Thật vậy, không mất tính tổng quát giả sử (x_n) hội tụ về giới hạn $L > 0$. Lấy giới hạn hai vế của công thức truy hồi ta được:

$$L = \frac{1}{2}\left(L + \frac{a}{L}\right) \Leftrightarrow 2L = L + \frac{a}{L}$$

$$\Rightarrow L = \frac{a}{L} \Rightarrow L^2 = a \Rightarrow \lim x_n = \sqrt{a} \Rightarrow \lim x_n^2 = a$$

Do đó dãy (x_n) tồn tại giới hạn.

Mặt khác, giả sử $x_0 = \frac{a}{2} > 0 \Rightarrow x_1 = \frac{1}{2}\left(x_0 + \frac{a}{x_0}\right) > 0$

Giả sử $x_n > 0$, ta có:

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$$

Do $x_n > 0 \Rightarrow \frac{a}{x_n} > 0 \Rightarrow x_{n+1} > 0$. Theo nguyên lý quy nạp, ta có:

$$x_n > 0 \quad \forall n \in \mathbb{N}$$

Mặt khác, áp dụng bất đẳng thức CauchySchwarz, ta có:

$$\frac{x_n + \frac{a}{x_n}}{2} \geq \sqrt{x_n \cdot \frac{a}{x_n}} = \sqrt{a} \Rightarrow x_{n+1} \geq \sqrt{a}$$

$$\Rightarrow x_n \geq \sqrt{a} \quad \forall n \in \mathbb{N} \tag{1}$$

Ta xét hiệu:

$$x_{n+1} - \sqrt{a} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) - \sqrt{a} = \frac{1}{2}\left(x_n - \sqrt{a} + \frac{a}{x_n} - \sqrt{a}\right)$$

Vì $x_n > \sqrt{a} \Rightarrow \frac{a}{x_n} < \sqrt{a}$, do đó cả hai vế đều nhỏ hơn $x_n - \sqrt{a}$, suy ra:

$$x_{n+1} - \sqrt{a} < x_n - \sqrt{a} \Rightarrow x_{n+1} < x_n \Rightarrow \text{Dãy giảm} \tag{2}$$

Từ (1) và (2), ta có điều phải chứng minh. (□)

Dùng vòng lặp `for`

```python
a = 10
x = a / 2  # giá ịtr ởkhi ạto
n = 10     # ốs vòng ặlp

for i in range(n):
    x = 0.5 * (x + a / x)

print(f"sqrt({a})  {x}")
```

Code Listing 5: Tìm căn bậc hai bằng Newton-Raphson với vòng lặp for

Dùng vòng lặp `while` với sai số `epsilon`

```python
a = 10
x = a / 2
epsilon = 1e-8

while True:
    next_x = 0.5 * (x + a / x)
    if abs(x - next_x) < epsilon:
        break
    x = next_x

print(f"sqrt({a})  {x}")
```

Code Listing 6: Tìm căn bậc hai bằng Newton-Raphson với vòng lặp while