AGENTRA

# Cấu trúc rẽ nhánh trong Python

Tin học · Lớp 10

Giáo viên: Phúc Hảo

# Mục tiêu bài học

**Ý chính:** Máy tính biết **"tự quyết định"** — chọn việc phải làm dựa trên một **điều kiện** đúng hay sai.

Sau tiết học này, em có thể:

- Hiểu và viết được câu lệnh `if`, `if -- else`, `if -- elif -- else` trong Python.
- Diễn đạt một **điều kiện** bằng biểu thức so sánh/logic đúng cú pháp.
- Viết chương trình rẽ nhánh giải một bài toán có điều kiện.
- **Kiểm thử** chương trình trên nhiều trường hợp để xác nhận đúng.

# Khởi động

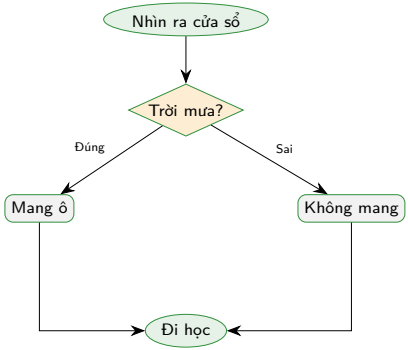
**Tình huống**

Sáng đi học, em nhìn ra cửa sổ và tự nhủ:

*"**Nếu** trời mưa **thì** mang ô, **còn không** thì thôi."*

Bộ não em vừa xử lí một **điều kiện**: nó **kiểm tra** "trời có mưa không?" rồi **chọn** hành động phù hợp.

**Câu hỏi:** Làm sao để máy tính cũng biết "cân nhắc rồi mới làm" như vậy? $\Rightarrow$ Đó là **cấu trúc rẽ nhánh**.



# 1

## Kiến thức: Câu lệnh rẽ nhánh

# Cấu trúc rẽ nhánh là gì?

**Định nghĩa**

**Cấu trúc rẽ nhánh** cho phép chương trình **kiểm tra một điều kiện**, rồi **chọn thực hiện** khối lệnh này hay khối lệnh khác tùy theo điều kiện **đúng** (`True`) hay **sai** (`False`).

- Trái với **cấu trúc tuần tự** (làm lần lượt từ trên xuống), rẽ nhánh có thể **bỏ qua** một số lệnh.
- Bộ phận quyết định gọi là **điều kiện** — một biểu thức chỉ nhận giá trị **Đúng** hoặc **Sai**.

**Liên hệ thực tế**

Giống người gác cổng: **nếu** có vé **thì** cho vào, **nếu không** thì mời ra. Một lần kiểm tra → một trong hai lối đi.

# Điều kiện và các toán tử so sánh

**Điều kiện** là biểu thức trả về `True` hoặc `False`. Thường được lập từ các **toán tử so sánh**:

| Kí hiệu | Ý nghĩa | Ví dụ |
|---|---|---|
| `==` | bằng | `x == 0` |
| `!=` | khác | `n != 5` |
| `> <` | lớn / bé hơn | `t > 37` |
| `>= <=` | lớn/bé bằng | `d >= 5` |

**Ghi nhớ**

**Lỗi thường gặp:** dùng `=` (phép gán) thay cho `==` (so sánh bằng) trong điều kiện.

Nối nhiều điều kiện bằng `and`, `or`, phủ định bằng `not`:

```
0 <= d <= 10 and d != 5
```

# Dạng 1 — Câu lệnh `if` (rẽ một nhánh)

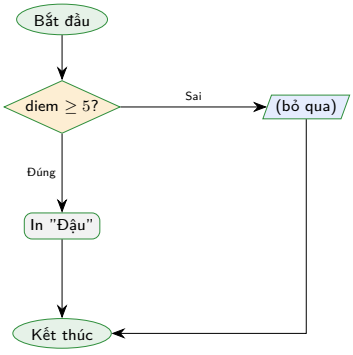
**Cú pháp**

```
if <điều kiện>:
  <khối lệnh>
```

- Điều kiện **Đúng** ⇒ thực hiện khối lệnh (được thụt vào trong).
- Điều kiện **Sai** ⇒ **bỏ qua**, chạy tiếp lệnh phía sau.

**Ví dụ**

```
if diem >= 5:
  print("Đậu")
```

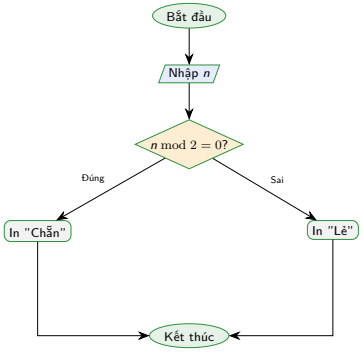


# Dạng 2 — Câu lệnh `if -- else` (rẽ hai nhánh)

## Cú pháp

```
if <điều kiện>:
    <khối 1>
else:
    <khối 2>
```

Điều kiện **Đúng** → khối 1; **Sai** → khối 2. **Luôn có đúng một** khối được chạy.




## Ghi nhớ

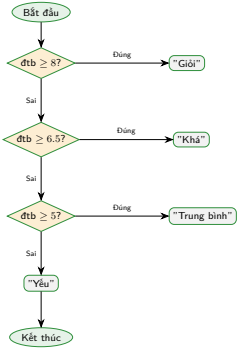
Đừng quên dấu **hai chấm** : sau `if` và `else`.

# Dạng 3 — `if -- elif -- else` (rẽ nhiều nhánh)

## Cú pháp

```
if <đk 1>:
   <khối 1>
elif <đk 2>:
   <khối 2>
else:
   <khối 3>
```

Các điều kiện được kiểm tra **lần lượt từ trên xuống**. Gặp điều kiện **Đúng** đầu tiên thì chạy khối đó rồi **dừng xét** phần còn lại.



# So sánh ba dạng rẽ nhánh

| Dạng | Số nhánh | Khi nào dùng | Ví dụ điển hình |
|---|---|---|---|
| `if` | 1 (làm / bỏ qua) | chỉ cần xử lí khi Đúng | Nếu điểm $\geq 5$ thì báo "Đậu" |
| `if -- else` | 2 | hai khả năng loại trừ nhau | Chẵn / Lẻ, Đậu / Rớt |
| `if -- elif -- else` | $\geq 3$ | nhiều trường hợp phân bậc | Xếp loại Giỏi/Khá/TB/Yếu |

**Ý chính:** Cùng một logic, chọn dạng **ít nhánh nhất đủ dùng** để chương trình gọn và dễ đọc.

**Ghi nhớ**

**Thụt lề (indentation)** quy định lệnh nào thuộc nhánh nào. Thụt lề sai $\Rightarrow$ Python báo `IndentationError`. Hãy dùng thống nhất **4 dấu cách** cho mỗi mức.

# 2

# Ví dụ minh hoạ

# Ví dụ 1 — Kiểm tra số chẵn hay lẻ

**Bài toán:** Nhập số nguyên $n$, in ra $n$ là số "Chẵn" hay "Lẻ".

- **Bước 1 — Xác định điều kiện.** Số chẵn là số chia hết cho 2, tức **số dư khi chia 2 bằng 0**: điều kiện là `n % 2 == 0`.
- **Bước 2 — Chọn cấu trúc.** Có **đúng hai** khả năng loại trừ nhau (chẵn / lẻ) $\Rightarrow$ dùng `if -- else`.
- **Bước 3 — Viết chương trình.**

**Chương trình**

```
n = int(input("Nhập n: "))
if n % 2 == 0:
  print("Chẵn")
else:
  print("Lẻ")
```

**Kết quả:** với `n = 8` in "Chẵn"; với `n = 7` in "Lẻ". *Vì sao?* $8 \bmod 2 = 0$ (Đúng) nên chạy nhánh `if`; $7 \bmod 2 = 1$ (Sai) nên chạy nhánh `else`.

# Ví dụ 2 — Xếp loại học lực theo điểm trung bình

**Bài toán:** Nhập điểm trung bình `dtb` (0–10), xếp loại: $\geq 8$ Giỏi, $\geq 6.5$ Khá, $\geq 5$ Trung bình, còn lại Yếu.

- **Bước 1 — Nhận diện nhiều trường hợp phân bậc.** Bốn mức loại trừ nhau $\Rightarrow$ dùng `if` -- `elif` -- `else`.
- **Bước 2 — Sắp điều kiện từ cao xuống thấp.** Vì Python dừng ở nhánh Đúng đầu tiên, nếu xét `>=5` trước thì điểm 9 cũng lọt vào "Trung bình" — **sai**. Phải xét `>=8` trước.

**Chương trình**

```
dtb = float(input("Nhập điểm TB: "))
if dtb >= 8: print("Giỏi")
elif dtb >= 6.5: print("Khá")
elif dtb >= 5: print("Trung bình")
else: print("Yếu")
```

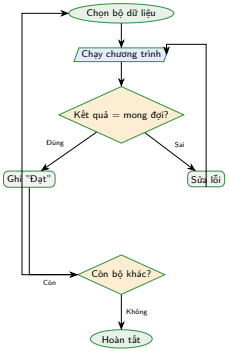
**Kết quả:** `dtb = 9` → "Giỏi"; `dtb = 6.5` → "Khá"; `dtb = 4` → "Yếu".

# Kiểm thử chương trình trên nhiều trường hợp

**Kiểm thử** là chạy thử chương trình với các **bộ dữ liệu khác nhau** để so kết quả thực tế với kết quả **mong đợi**.

Nên chọn cả các **giá trị biên** — chỗ điều kiện "vừa đổi nhánh" — vì lỗi hay nấp ở đó.

| `dtb` | **Mong đợi** | **Ghi chú** |
|---|---|---|
| 9.0 | Giỏi | giữa nhánh |
| 8.0 | Giỏi | **biên** $= 8$ |
| 6.5 | Khá | **biên** $= 6.5$ |
| 4.9 | Yếu | sát dưới $5$ |



# 3

## Luyện tập nhanh

## Luyện tập nhanh · Số lớn hơn

Nhập hai số `a`, `b`; in ra số lớn hơn (nếu bằng nhau in "Bằng nhau"). Em chọn dạng nào và điều kiện đầu tiên là gì?

**A.** `if` một nhánh, điều kiện `a == b`

**B.** `if` -- `elif` -- `else`, điều kiện đầu `a > b`

**C.** chỉ cần `if` -- `else` là đủ mọi trường hợp

**Gợi ý đáp án**

**Đáp án B.** Có **ba** khả năng ($a > b$, $a < b$, $a = b$) nên dùng `if` -- `elif` -- `else`:

```
if a > b: ...  elif a < b: ...  else: print("Bằng nhau")
```

## Luyện tập nhanh · Bắt lỗi thứ tự điều kiện

Một bạn viết chương trình xếp loại nhưng đặt `if dtb >= 5` **lên trên** `elif dtb >= 8`. Bộ dữ liệu nào sẽ **lộ ra lỗi** khi kiểm thử?

**A.** `dtb = 4` **B.** `dtb = 9` **C.** `dtb = 5`

**Gợi ý đáp án**

**Đáp án B.** Với `dtb = 9`, chương trình gặp `>= 5` Đúng **trước** nên in "Trung bình" thay vì "Giỏi". **Bài học:** luôn xếp điều kiện **từ chặt đến lỏng** và kiểm thử ở giá trị lớn.

# Luyện tập nhanh - Câu 1

## Luyện tập nhanh · Nhận biết

**Câu 1:** Câu lệnh rẽ nhánh dạng thiếu (chỉ có `if`) trong Python được viết *đúng cú pháp* là:

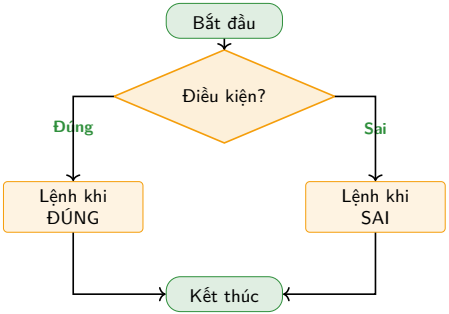
**A.** `if x > 0 then:`
**B.** `if x > 0:`
**C.** `if (x > 0)`
**D.** `if x = 0:`

# Đáp án - Câu 1

**Đáp án đúng: B.** Câu lệnh `if` kết thúc bằng dấu hai chấm `:` rồi mới đến khối lệnh thụt vào.

**Vì sao các phương án kia là bẫy:** A thêm `then` (đó là cú pháp ngôn ngữ khác, Python không có); C *quên dấu* `:` — lỗi rất hay gặp; D dùng `=` (phép gán) thay vì `==` khi so sánh.

# Luyện tập nhanh - Câu 2




## Luyện tập nhanh · Nhận biết

**Câu 2:** Theo sơ đồ khối bên, khi *điều kiện sai* thì chương trình thực hiện nhánh nào?

**A.** Nhánh "Lệnh khi ĐÚNG"
**B.** Nhánh "Lệnh khi SAI"
**C.** Cả hai nhánh
**D.** Không nhánh nào

# Đáp án - Câu 2

**Đáp án đúng: B.** Điều kiện sai thì rẽ theo nhánh "Sai" (phần `else`).

**Vì sao các phương án kia là bẫy:** A là nhánh chạy khi điều kiện *đúng* (dễ nhầm ngược); C sai vì mỗi lần chỉ đi *một* nhánh; D sai vì cấu trúc `if-else` luôn thực hiện một trong hai nhánh.

# Luyện tập nhanh - Câu 3

Luyện tập nhanh · Nhận biết

**Câu 3:** Trong Python, từ khóa nào dùng cho nhánh "ngược lại" (thực hiện khi điều kiện của `if` sai)?

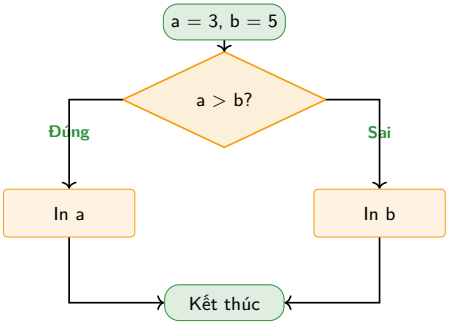
**A.** `elif`
**B.** `else`
**C.** `otherwise`
**D.** `then`

# Đáp án - Câu 3

**Đáp án đúng: B.** `else` là nhánh chạy khi điều kiện của `if` không thỏa mãn.

**Vì sao các phương án kia là bẫy:** A `elif` nghĩa là "nếu không thì nếu..." — dùng để *thêm* một điều kiện, không phải nhánh cuối; C và D không phải từ khóa của Python (là cú pháp ngôn ngữ khác).

# Luyện tập nhanh - Câu 4




## Luyện tập nhanh · Thông hiểu

**Câu 4:** Với `a = 3`, `b = 5`, sơ đồ bên sẽ in ra giá trị nào?

**A.** 3
**B.** 5
**C.** 8
**D.** Không in gì

# Đáp án - Câu 4

**Đáp án đúng: B (in ra 5).** Vì 3 > 5 là *sai* nên rẽ nhánh "Sai" $\rightarrow$ In b $= 5$.

**Vì sao các phương án kia là bẫy:** A (in 3) là do nhầm luôn chạy nhánh "Đúng"; C (=8) là nhầm sang phép cộng a+b; D sai vì luôn có đúng một nhánh được in.

# Luyện tập nhanh - Câu 5

Luyện tập nhanh · Thông hiểu

**Câu 5:** Đoạn chương trình sau in ra kết quả gì?

```
x = 7
if x % 2 == 0:
    print("Chan")
else:
    print("Le")
```

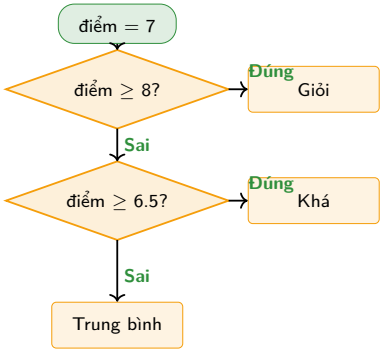
**A.** Chan **B.** Le **C.** 7 **D.** Báo lỗi

# Đáp án - Câu 5

**Đáp án đúng: B (in ra "Le").** Vì `7 % 2 = 1` khác 0 nên điều kiện sai, chạy nhánh `else`.

**Vì sao các phương án kia là bẫy:** A do nhầm `7 % 2 == 0` là đúng; C do tưởng lệnh in ra biến `x`; D sai vì cú pháp hoàn toàn hợp lệ.

# Luyện tập nhanh - Câu 6




## Luyện tập nhanh · Thông hiểu

**Câu 6:** Chạy sơ đồ `if-elif-else` bên với `điểm = 7`, kết quả xếp loại là?

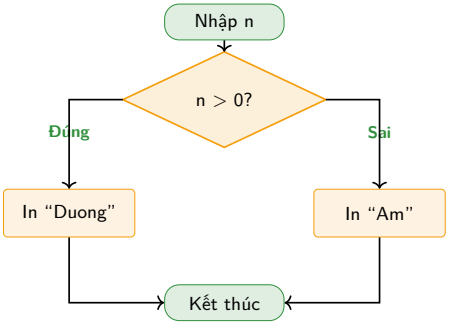
**A.** Giỏi
**B.** Khá
**C.** Trung bình
**D.** Không xác định

# Đáp án - Câu 6

**Đáp án đúng: B (Khá).** `7` $\geq$ `8` sai $\rightarrow$ xét tiếp `7` $\geq$ `6.5` đúng $\rightarrow$ "Khá".

**Vì sao các phương án kia là bẫy:** A do nhầm `7` $\geq$ `8`; C do bỏ qua nhánh `elif` mà nhảy thẳng xuống `else`; D sai vì cấu trúc luôn cho ra đúng một xếp loại.

# Luyện tập nhanh - Câu 7




## Luyện tập nhanh · Vận dụng

**Câu 7:** Chương trình trên xét dấu của `n` nhưng *thiếu* trường hợp `n = 0`. Cần chọn bộ dữ liệu **kiểm thử** nào để lộ ra lỗi này?

**A.** $n = 5$
**B.** $n = -3$
**C.** $n = 0$
**D.** $n = 100$

# Đáp án - Câu 7

**Đáp án đúng: C (n = 0).** Với `n = 0`, điều kiện `n > 0` sai nên máy in "Am" — sai kết quả, làm lộ lỗi thiếu nhánh.

**Vì sao các phương án kia là bẫy:** A và D (số dương) cho kết quả đúng; B (số âm) cũng đúng — cả ba đều *không* chạm tới trường hợp biên `n = 0`, nên không phát hiện được lỗi.

Luyện tập nhanh · Vận dụng

**Câu 8:** Cần viết điều kiện kiểm tra "`n` là số chẵn *và* dương". Biểu thức nào đặt sau `if` là đúng?

**A.** `n % 2 == 0 and n > 0`
**B.** `n % 2 == 0 or n > 0`
**C.** `n % 2 = 0 and n > 0`
**D.** `n > 0 and n % 2 == 1`

# Đáp án - Câu 8

**Đáp án đúng: A.** Cần đồng thời chẵn (`n % 2 == 0`) *và* dương (`n > 0`) nên dùng `and`.

**Vì sao các phương án kia là bẫy:** B dùng `or` → chỉ cần một điều kiện đúng, sai logic; C dùng `=` (gán) thay vì `==`; D nhầm `% 2 == 1` là số *lẻ*, không phải chẵn.

# Nội dung luyện tập ở lớp

1. **Bài 1.** Nhập một số nguyên `n`, in ra "So duong", "So am" hoặc "So khong" (dùng `if-elif-else`).
2. **Bài 2.** Nhập hai số `a`, `b`; in ra số lớn hơn. Kiểm thử với `a>b`, `a<b` và `a=b`.
3. **Bài 3.** Nhập điểm trung bình, xếp loại: $\geq 8$ Giỏi, $\geq 6.5$ Khá, $\geq 5$ Trung bình, còn lại Yếu.

## Gợi ý

Vẽ sơ đồ khối trước khi viết code. Nhớ dấu `:` cuối dòng `if/elif/else` và *thụt lề* khối lệnh. Luôn kiểm thử cả trường hợp **biên** (bằng nhau, bằng 0) để không sót nhánh.

# Bài tập về nhà

1. Nhập `n`, kiểm tra `n` chẵn hay lẻ và in kết quả.
2. Nhập năm, kiểm tra có phải năm nhuận không (chia hết cho 4).
3. Giải và biện luận phương trình bậc nhất `ax + b = 0`.
4. Nhập 3 số, tìm và in ra số lớn nhất.
5. Nhập số tiền mua hàng, tính giảm giá: $\geq$ 500k giảm 10%, còn lại không giảm.

**Đọc trước bài sau:** "Cấu trúc lặp `for` trong Python" — tìm hiểu ý nghĩa của `for` và `range()`.

# Bảng tổng hợp cú pháp rẽ nhánh

| Dạng | Cú pháp rút gọn | Khi dùng |
|---|---|---|
| Thiếu (if) | `if <đk>:` | Chỉ xử lý khi điều kiện đúng |
| Đủ (if-else) | `if <đk>: …else:` | Có việc cho cả đúng và sai |
| Nhiều nhánh | `if …elif …else:` | Phân loại theo nhiều mức |

**Ghi nhớ:** mỗi dòng `if/elif/else` kết thúc bằng dấu `:`; khối lệnh bên trong phải *thụt lề* thẳng hàng; so sánh dùng `==`, gán dùng `=`.

# Bảng gợi ý kiểm thử chương trình

| Bài toán | Trường hợp cần thử | Mục đích |
|---|---|---|
| Xét dấu `n` | `n>0`, `n<0`, `n=0` | Không sót nhánh biên |
| So sánh `a`, `b` | `a>b`, `a<b`, `a=b` | Kiểm tra trường hợp bằng |
| Xếp loại điểm | 9, 7, 5, 3 | Phủ mọi mức phân loại |

**Nguyên tắc:** chọn dữ liệu thử *đại diện* cho từng nhánh và đặc biệt chú ý các giá trị **ở ranh giới** — đó là nơi lỗi hay ẩn nấp.

# Tổng kết

- **Ba dạng rẽ nhánh:** `if` (một nhánh), `if-else` (hai nhánh), `if-elif-else` (nhiều nhánh) — chọn dạng theo số trường hợp cần xử lý.
- **Cú pháp chuẩn:** kết thúc dòng điều kiện bằng `:`, thụt lề khối lệnh; phân biệt `==` (so sánh) với `=` (gán).
- **Điều kiện** là biểu thức logic cho giá trị Đúng/Sai; có thể ghép bằng `and`, `or` — nhớ đúng ý nghĩa của mỗi phép.
- **Kiểm thử** nhiều trường hợp, nhất là giá trị biên, để chương trình không sót nhánh và cho kết quả chính xác.

*Liên hệ: mọi quyết định trong đời sống — rẽ trái hay phải, mang áo mưa hay không — đều là một cấu trúc rẽ nhánh; lập trình chỉ là cách diễn đạt lại tư duy đó cho máy tính hiểu.*

# Cảm ơn các em đã lắng nghe!

Bài giảng biên soạn với EduFlow · agentra.io.vn