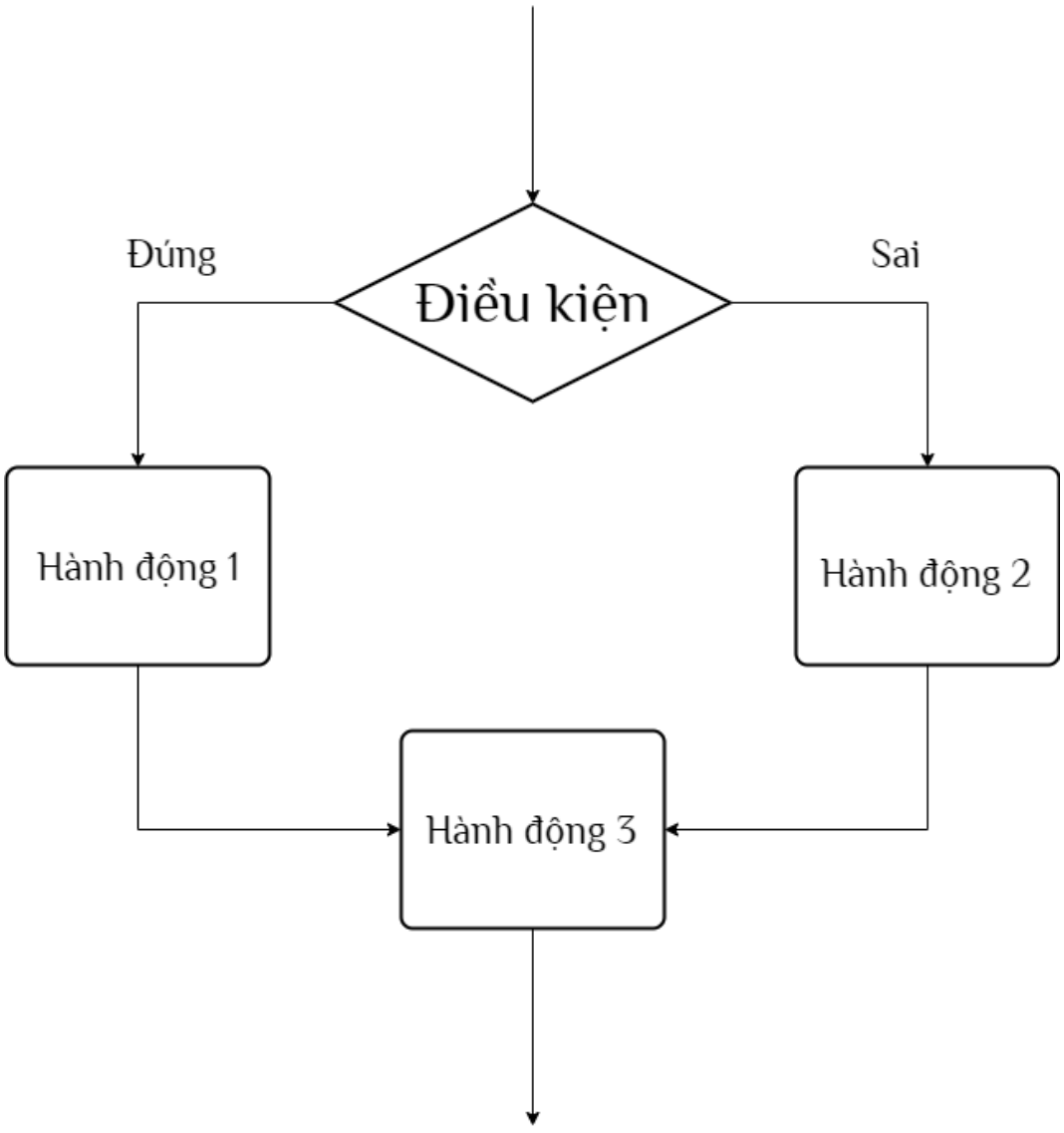
Cú pháp rẽ nhánh trong python

December 18, 2020

1 Cấu trúc rẽ nhánh cơ bản

1.1 Giới thiệu

Dạng đầy đủ của cấu trúc rẽ nhánh được thể hiện qua sơ đồ sau:



Đầu tiên, chúng ta sẽ kiểm tra điều kiện :

- Nếu điều kiện đúng, chúng ta thực hiện **hành động 1** rồi thực hiện tiếp **hành động 3**.
- Nếu điều kiện sai, chúng ta thực hiện **hành động 2** rồi thực hiện tiếp **hành động 3**

1.2 Cách viết trong python

Để viết câu lệnh rẽ nhánh trong python, chúng ta viết như sau :

```
if <điều_kiện> :
    <hành_động_khi_đúng>
else :
    <hành_động_khi_sai>
```

```
In [5]: # tạo biến a
        a = int(input('Nhập a '))
        # cấu trúc rẽ nhánh
        if a >= 0:
          print('Đây là số không âm.')
          print('')
        else: # a < 0
          print('Đây là số âm.')

        print('Dù đúng hay sai, câu này luôn được in.')
```

```
Nhập a 123
Đây là số không âm.

Dù đúng hay sai, câu này luôn được in.
```

Bạn có thể bỏ qua phần `else` nếu như không có hành động gì. Lúc này, ta sẽ có dạng *không đầy đủ* của cấu trúc rẽ nhánh.

```
In [6]: b = -2
        if b > 0 :
          print('Đây là số dương')
        print('Dù đúng hay sai, câu này luôn có!')
```

```
Dù đúng hay sai, câu này luôn có!
```

Bài tập : Viết chương trình nhập vào một số. Nếu là số chẵn thì in ra `'Đây là số chẵn'`, ngược lại, nếu là số lẻ thì in ra `'Đây là số lẻ'`.

2 Cấu trúc rẽ nhánh nâng cao

2.1 Cấu trúc rẽ nhánh lồng nhau

Bên trong các `hành_động` ở trên, chúng ta hoàn toàn có thể sử dụng các cấu trúc rẽ nhánh khác. Người ta gọi đó là cấu trúc rẽ nhánh lồng nhau.

```
In [10]: # ví dụ :
         a = 6
         if a >= 5 :
           if a > 8.5 :
```

```
            print('Học sinh giỏi')
          else: # a < 8.5
            if a > 7 :
              print('Học sinh khá')
        else :
          print('Chúc bạn may mắn lần sau')
```

2.2 Cấu trúc rẽ nhánh bậc thang

Đây là một trường hợp đặc biệt của dạng lồng nhau, khi bạn kiểm tra lần lượt từng điều kiện. Tức là, bạn kiểm tra điều kiện 1, nếu điều kiện 1 sai, bạn kiểm tra điều kiện 2, nếu lại sai, bạn kiểm tra điều kiện 3, cứ thế đến điều kiện cuối cùng.

Cấu trúc rẽ nhánh bậc thang được viết trong python như sau :

```
if <điều_kiện_1> :
    <hành_động_1>
elif <điều_kiện_2> :
    <hành_động_2>
elif <điều_kiện_3> :
    <hành_động_3>
...
else :
    <hành_động_gì_đó>
```

Lưu ý : - Số lượng `elif` là không giới hạn. - Có thể không có `else` hoặc có một `else`.

```
In [5]: # ví dụ
        dt = float(input('Nhập điểm trung bình '))
        if dt > 8 :
          print('Học sinh giỏi')
        elif dt > 6.5 :
          print('Học sinh khá')
        elif dt > 5 :
          print('Học sinh trung bình')
        elif dt > 2 :
          print('Học sinh yếu')
        else :
          print('Học sinh kém')
```

```
Nhập điểm trung bình 2
Học sinh kém
```

2.3 Cấu trúc rẽ nhánh `switch-case`

Đây là một dạng của cấu trúc rẽ nhánh bậc thang nói trên, khi mà bạn thực hiện hành động dựa trên giá trị của một biến số nào đó.

```
In [0]: # ví dụ :
        a = int(input('Nhập vào một số '))
        if a == 1 :
```

```
    print('Bạn nhập số 1')
elif a == 2 :
    print('Bạn nhập số 2')
else :
    print('Số này lớn quá')
```

Trong nhiều ngôn ngữ lập trình, người ta sử dụng các từ khoá `switch`, `case` để biểu diễn cấu trúc này.

Tuy nhiên, python lại **không hỗ trợ** các từ khoá đó. Để viết cấu trúc `switch-case`, bạn có thể dùng dạng rẽ nhánh bậc thang như minh hoạ trên hoặc dùng đến `dictionary` (sẽ nhắc ở bài khác).

3 Điều kiện trong cấu trúc rẽ nhánh

Điều kiện trong cấu trúc rẽ nhánh có đặc điểm sau : - Phải có giá trị là `True` hoặc `False`. - Có thể là biểu thức đơn hoặc ghép, gọi chung là biểu thức logic

3.1 Biểu thức logic đơn

Là biểu thức chỉ gồm 1 phép toán so sánh.

Các phép toán so sánh trong python gồm : - > (lớn hơn), < (bé hơn), >= (lớn hơn hoặc bằng), <= (nhỏ hơn hoặc bằng) - == (so sánh bằng), != (so sánh khác)

```
In [11]: 5 != 3
```

```
Out[11]: True
```

```
In [13]: c = int(input('Nhập c '))
         t = (c > 2)
         print(t)
```

```
Nhập c 10
True
```

3.2 Biểu thức logic ghép

Là các biểu thức đơn được kết nối với nhau bằng dấu ngoặc `()` và các toán tử logic.

Các toán tử logic trong python gồm :

- `and` : phép và (phép hợp)
- `or` : phép hoặc (phép tuyển)
- `not` : phép phủ định

```
In [13]: (1 > 2) and (3 != 5)
```

```
Out[13]: False
```

```
In [14]: (1 > 2) or (3 != 5)
```

```
Out[14]: True
```

```
In [16]: not (1 > 2)
```

```
Out[16]: True
```

3.3 Một số điểm riêng của python

Python có quy ước các giá trị 0, '' (chuỗi rỗng), None được ngầm hiểu là False. Tuy nhiên, điều này chỉ áp dụng cho điều kiện trong câu rẽ nhánh.

```
In [17]: not 0

Out[17]: True

In [18]: not ''

Out[18]: True

In [19]: not None

Out[19]: True
```

Python cho phép bạn viết một số biểu thức logic dưới dạng rút gọn. Ví dụ `(1 < a) and (a < 2)` có thể viết thành `1 < a < 2`.

```
In [8]: e = int(input('Nhập e '))
        1 < e < 5

Nhập e 4


Out[8]: True
```

4 Phép gán có điều kiện

Giả sử chúng ta có một biến x, chúng ta muốn gán giá trị cho x tuỳ thuộc một điều kiện nào đó. Về cơ bản, chúng ta có thể viết như sau

```
if <điều_kiện> :
    x = <giá_trị_khi_điều_kiện_đúng>
else :
    x = <giá_trị_khi_điều_kiện_sai>
```

Trong nhiều ngôn ngữ lập trình, người ta thay thế đoạn trên bằng 1 cú pháp gán đặc biệt được gọi là phép gán có điều kiện.

Phép gán có điều kiện trong python được viết như sau:

```
<tên_biến> = <giá_trị_khi_đúng> if <điều_kiện> else <giá_trị_khi_sai>
```

và đọc hiểu là `tên_biến` sẽ có `giá_trị_khi_đúng` nếu `điều_kiện` được thoả, còn không sẽ có `giá_trị_khi_sai`.

```
In [14]: # ví dụ
         d = 1 if (1 > 2) else 2
         print(d)

2
```