

Phân Tích Dữ Liệu Thực Tế với Python

Bài 2.1: Lập Trình Python Cơ Bản - P1



Quang-Khai Tran, Ph.D
CyberLab, 02/2023



(Ảnh: Internet)

Giảng viên



Quang-Khai Tran



Postdoctoral Scholar at **KISTI**
한국과학기술정보연구원



Studied Big Data Analytics at
**Korea University of Science
and Technology**

Facebook: <https://www.facebook.com/tgkhai2705/>

Email: tgkhai0527@gmail.com



De-Thu Huynh

Giảng viên thỉnh giảng: TS. Huỳnh Đệ Thủ

Facebook: <https://www.facebook.com/dethu.huynh>

Trợ giảng



Nguyễn Bùi Hoàng Long

Facebook:

<https://www.facebook.com/hoanglong.nguyenbui.96>



Nguyễn Trường Thuận

Facebook:

<https://www.facebook.com/truongthuannn>



Trần Phi Long

Facebook:

<https://www.facebook.com/IsaacFA1992>

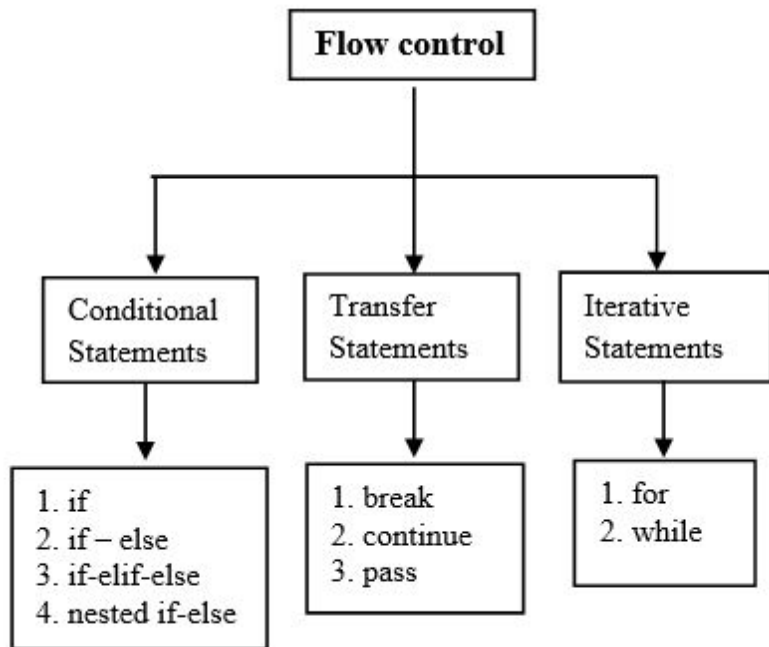
Nội dung



Lập trình Python cơ bản - Phần 1

1. Câu điều kiện
2. Vòng lặp
3. Tư duy lập trình
4. Bài tập
5. Thảo Luận & Hỏi đáp

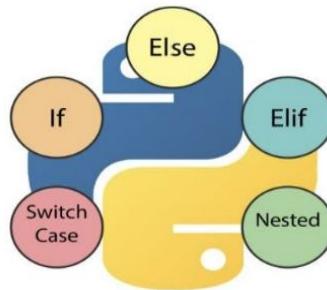
Luồng điều khiển trong Python





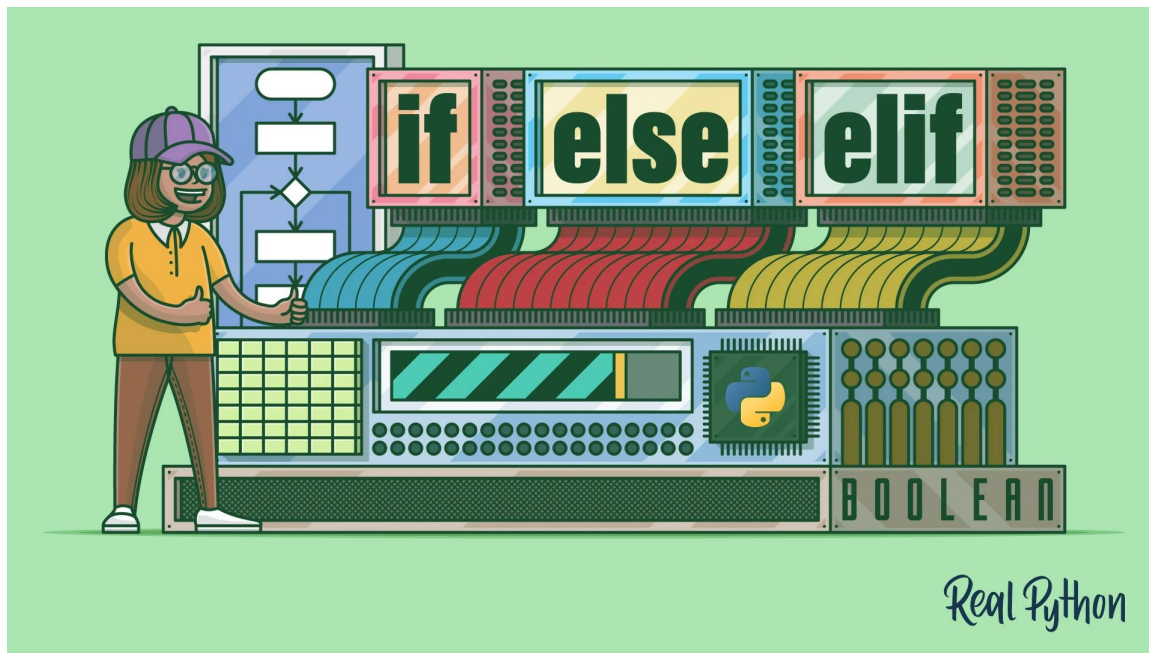
Phần 1. Câu điều kiện

Python conditional statements



learnbay.co

Conditional statement: dùng để phân chia các trường hợp theo điều kiện



Câu điều kiện (conditional statement: với các từ khóa **if elif else**) dùng để ra quyết định (decision making) chọn thực hiện lệnh thỏa mãn điều kiện cho trước

Câu điều kiện: if elif else

```
def Tính_Mức_KM(tuoi):  
    if tuoi < 5: km = 0.15  
    elif tuoi < 10: km = 0.10  
    else: km = 0.5  
  
    return km
```

```
km1 = Tính_Mức_KM(7)  
print(km1)
```

```
km2 = Tính_Mức_KM(3)  
print(km2)
```

0.1

0.15

Các phép luận lý (logical) cho điều kiện

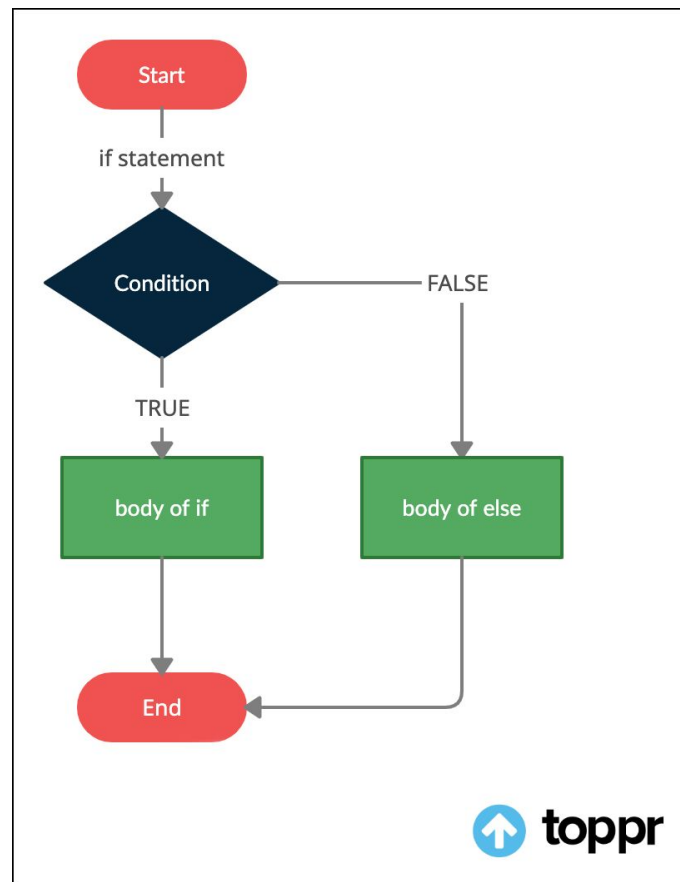
```
def Tính_Mức_KM(tuoi):  
    if type(tuoi) is not int:  
        print('Tuổi nhập vào không phải là số tự nhiên')  
        return None  
    elif not tuoi > 0:  
        print('Tuổi nhập vào phải là số tự nhiên lớn hơn 0')  
        return None  
  
    if tuoi < 5 or tuoi > 60: km = 0.15  
    elif 5 <= tuoi <= 10: km = 0.10  
    elif tuoi > 10 and tuoi <= 18: km = 0.75  
    else: km = 0.05  
  
    return km  
  
km1 = Tính_Mức_KM(70)  
print(km1)  
  
km2 = Tính_Mức_KM(-13)  
print(km2)
```

0.15

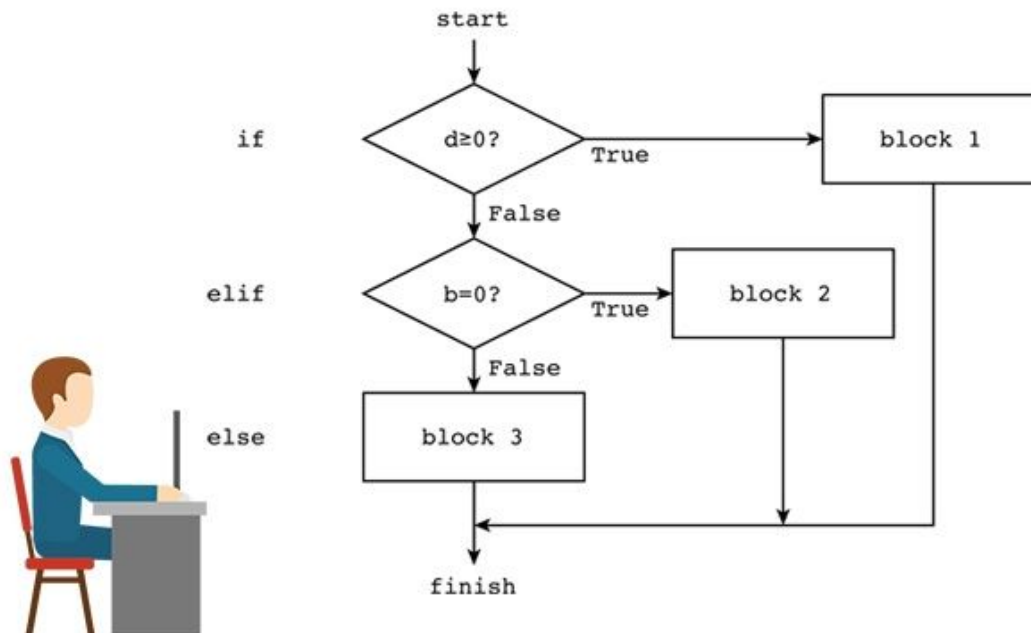
Tuổi nhập vào phải là số tự nhiên lớn hơn 0

None

- Lệnh **if** (nếu): kiểm tra nếu điều kiện thỏa
⇒ thực hiện khối lệnh trong **if**
- Lệnh **else** (khác): điều kiện không th
⇒ thực hiện khối lệnh trong **else**



- Lệnh **elif** (khác-nếu):
là kết hợp của else và if để thay thế việc phải viết thêm if sau else

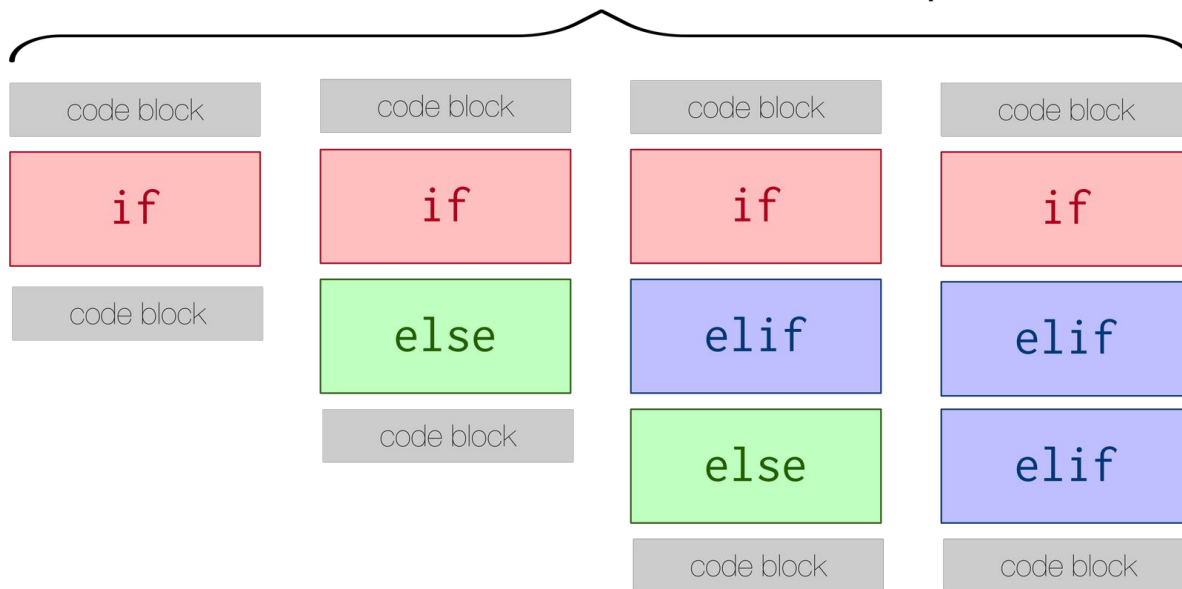


Trong câu điều kiện có thể:

- Tạo các câu điều kiện khác (nested conditional statement)
- Kết hợp nhiều điều kiện trong một lệnh **if/elif**
- Thực hiện trả về của một hàm bằng lệnh **return**
- Lưu ý: lệnh **else** là điều kiện không thỏa của **if/elif** đồng cấp gần nhất

Chú ý khi viết các câu if/elif/else

valid if/elif/else order examples



Switch-Case trong Python?

⇒ Ban đầu, Python không hỗ trợ (vì ko quá quan trọng, và có giải pháp khác)

```
In [7]: num=int(input("Enter a number : "))
if num==1:
    print("Monday")
elif num==2:
    print("Tuesday")
elif num==3:
    print("Wednesday")
elif num==4:
    print("Thursday")
elif num==5:
    print("Friday")
elif num==6:
    print("Saturday")
elif num==7:
    print("Sunday")
else:
    print("Please enter a number between 1 and 7")
```

But In Python 3.10 and after that, Python will support this:

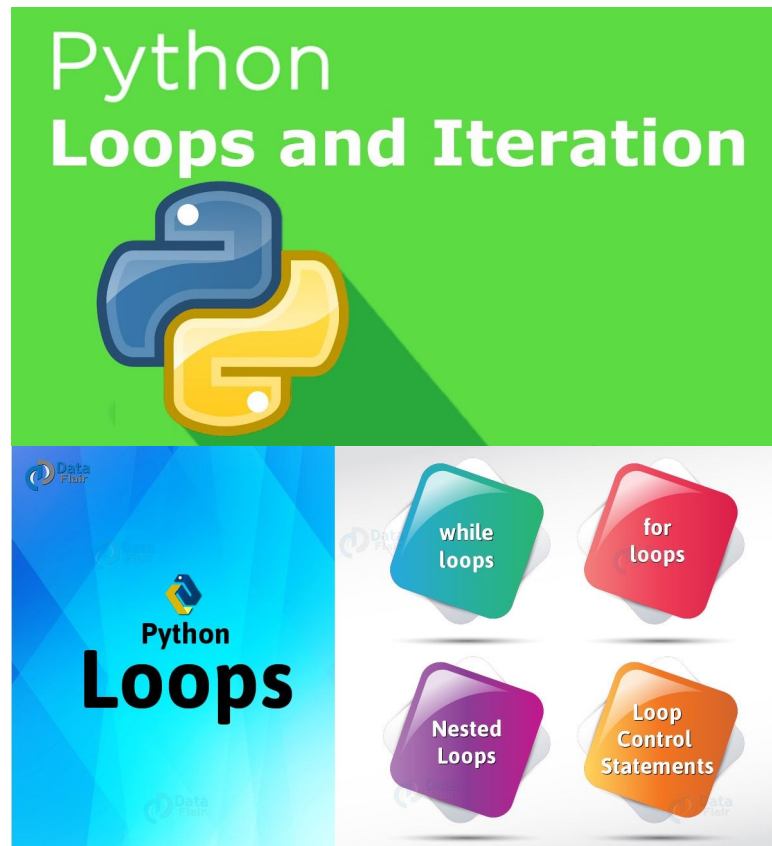
it's my example code:

Python3

```
def number_to_string(argument):
    match argument:
        case 0:
            return "zero"
        case 1:
            return "one"
        case 2:
            return "two"
        case default:
            return "something"

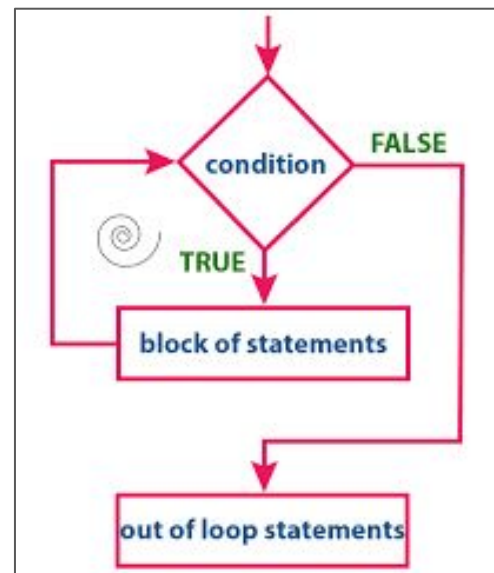
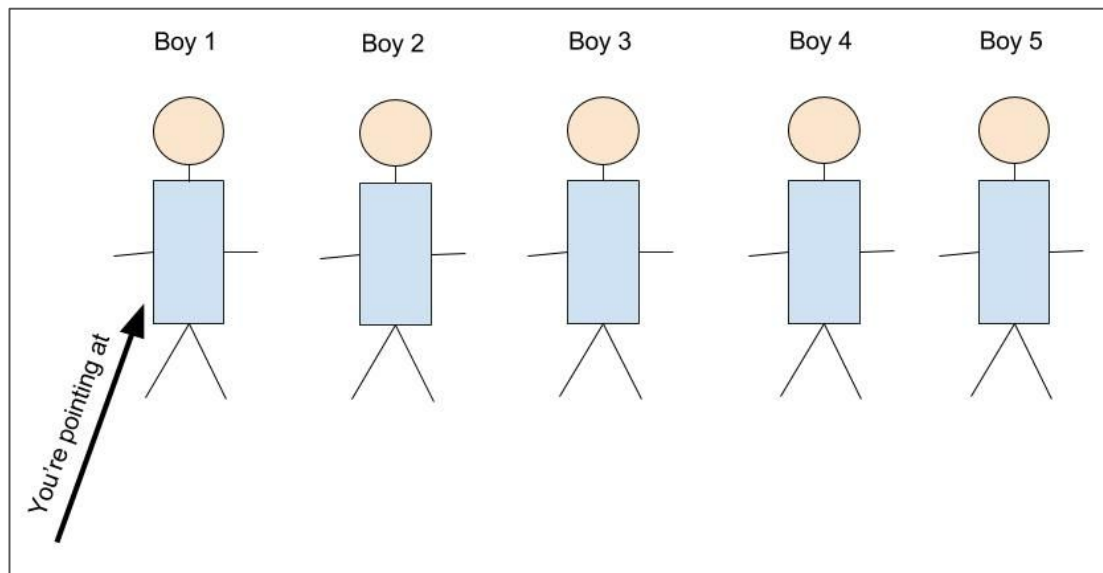
if __name__ == "__main__":
    argument = 0
    number_to_string(argument)
```

Phần 2. Vòng lặp



Trong lập trình cũng như trong cuộc sống:

⇒ có những việc phải thực hiện đi thực hiện lại



Vòng lặp là gì? Tại sao phải dùng vòng lặp?

⇒ Vòng lặp dùng để thực hiện/gọi một khối lệnh nhiều lần một cách lặp lại

⇒ Có 2 loại vòng lặp chính: **for** và **while**

```
danh_sach_hv = ["Nam", "Oanh", "Peter", "Laura"]  
for hv in danh_sach_hv:  
    print("Xin chào bạn", hv)
```

```
Xin chào bạn Nam  
Xin chào bạn Oanh  
Xin chào bạn Peter  
Xin chào bạn Laura
```

Vòng lặp for: Thực hiện lặp đi lặp lại một đoạn mã với một số lần nào đó

```
### Vòng lặp FOR (1)
danh_sach_hv = ["Nam", "Oanh", "Peter", "Laura"]
for hv in danh_sach_hv: # Lấy trực tiếp từng item
    print("Xin chào bạn", hv)
```

```
Xin chào bạn Nam
Xin chào bạn Oanh
Xin chào bạn Peter
Xin chào bạn Laura
```

```
### Vòng lặp FOR (2)
danh_sach_hv = ["Nam", "Oanh", "Peter", "Laura"]
for i in range(4): # Lấy index của item
    print("Xin chào bạn", danh_sach_hv[i])
```

```
Xin chào bạn Nam
Xin chào bạn Oanh
Xin chào bạn Peter
Xin chào bạn Laura
```

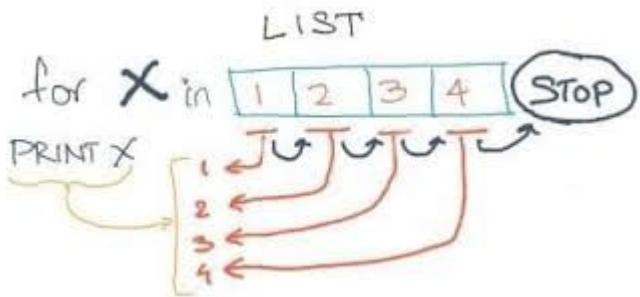
Vòng lặp while: Thực hiện lặp đi lặp lại một đoạn mã khi các điều kiện cho trước vẫn đúng

```
### Vòng lặp WHILE
x = 1
while x < 10:
    y = 2**x #2 mũ x
    print(f"x = {x}, y = {y}")
    x += 1
```

```
x = 1, y = 2
x = 2, y = 4
x = 3, y = 8
x = 4, y = 16
x = 5, y = 32
x = 6, y = 64
x = 7, y = 128
x = 8, y = 256
x = 9, y = 512
```

Vòng lặp for

- Duyệt qua các phần tử trong một danh sách, từ đầu đến cuối

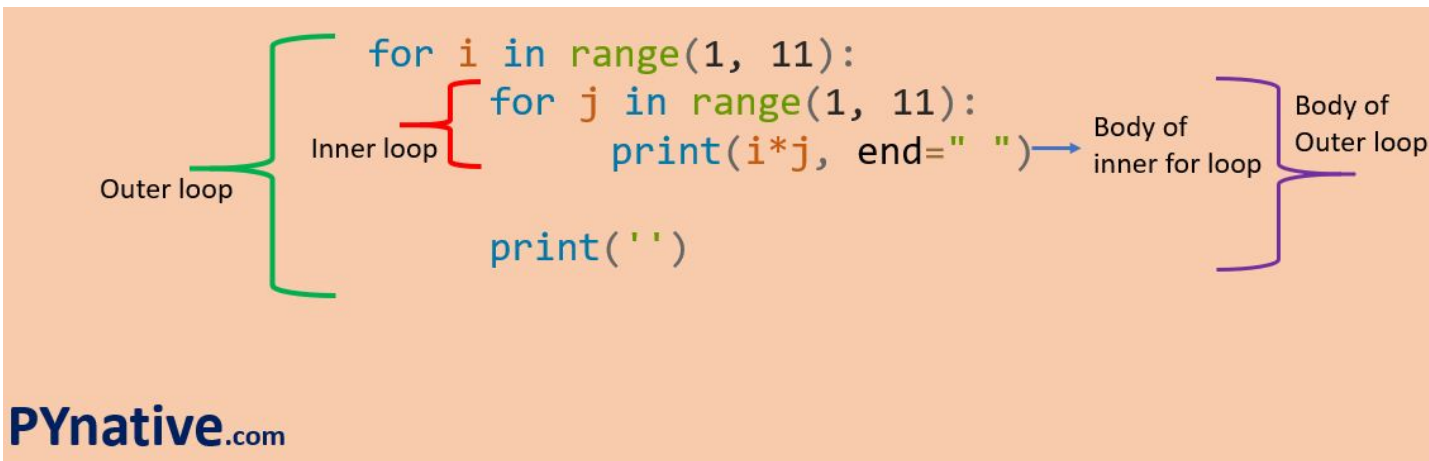


Vòng lặp while: thực hiện lặp lại cho đến khi một điều kiện vẫn thỏa

```
while Condition:  
    statement 1  
    statement 2  
    ...  
    statement n  
else:  
    statement(s)
```



Vòng lặp lồng nhau (nested loop)

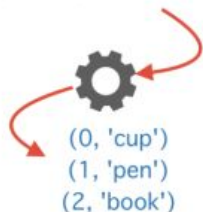


Vòng lặp for với enumerate:

Cho phép lặp và lấy thêm thành phần chỉ số ở trước mỗi item

Python Enumerate

['cup', 'pen', 'book']



Don't Write This:

```
my_container = ['Larry', 'Moe', 'Curly']  
index = 0  
for element in my_container:  
    print ('{} {}'.format(index, element))  
    index += 1
```

Write This:

```
my_container = ['Larry', 'Moe', 'Curly']  
for index, element in enumerate(my_container):  
    print ('{} {}'.format(index, element))
```

Vòng lặp for với enumerate: cú pháp

```
for index, item in enumerate(danh_sach, start = 0):  
    ... <statements>
```

enumerate(iterable, start=0)

Generate (counter, element) pairs for each element in iterable, starting to count from start (per default start=0).

```
fruits = ['apple', 'banana', 'cherry']
```

```
for i in range(len(fruits)):  
    print(i, fruits[i])
```

```
# Output:  
# 0 apple  
# 1 banana  
# 2 cherry
```



Not Pythonic

```
fruits = ['apple', 'banana', 'cherry']
```

```
for i, fruit in enumerate(fruits):  
    print(i, fruit)
```

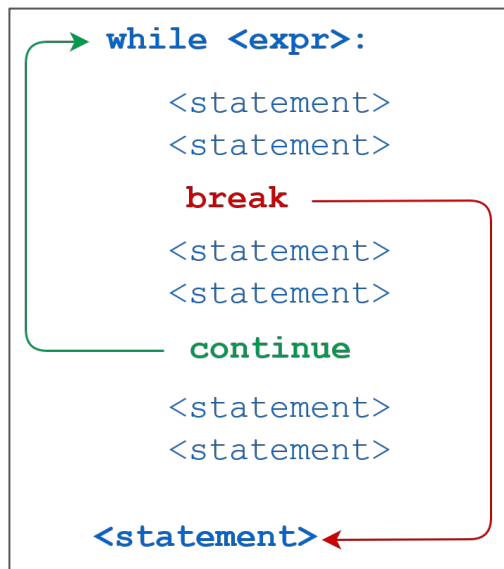
```
# Output:  
# 0 apple  
# 1 banana  
# 2 cherry
```



Pythonic

Thay đổi vòng lặp giữa chừng với **break** hoặc **continue**:

- Break: dừng luôn vòng lặp
- Continue: tạm thời không thực hiện các xử lý với biến đếm hiện tại, nhảy qua biến đếm tiếp theo
- Dừng cho cả **for/while**



Phần 3. Tư duy lập trình



Tư duy lập trình là gì?

Khi gặp một vấn đề (bài toán, yêu cầu), ta cần suy nghĩ để tìm cách giải quyết trước khi viết code



Programming is Thinking,
Not Typing.

Mục tiêu: tìm ra cách giải quyết tốt nhất, phương án thích hợp nhất

Tư duy lập trình là gì?

Một số thuật ngữ tiếng Anh:

- Logical thinking (for programming)
- Programming thinking
- Logic in programming

Essential Soft Skills Every Programmer Needs



LOGICAL

TECHNO**Kids**

<https://cyberlab.edu.vn/>





Một số practices thường được sử dụng khi giải quyết vấn đề:

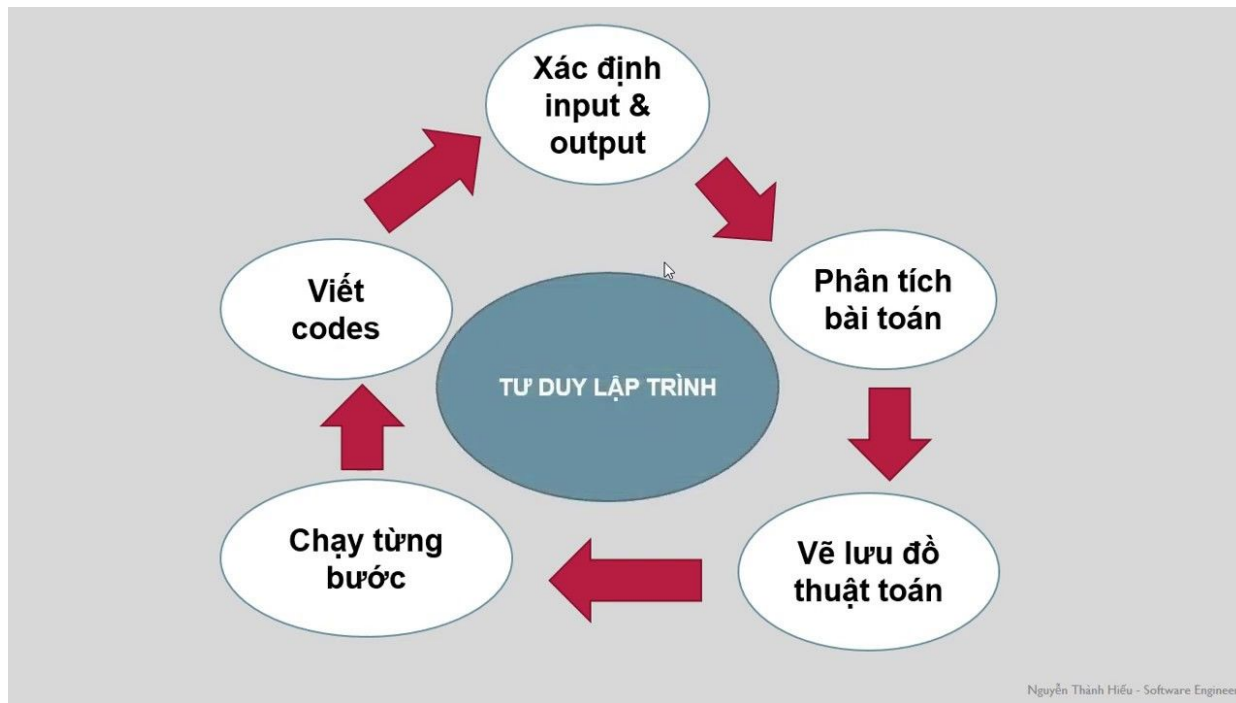
- Xem xét các thông tin nào là quan trọng, thông tin nào có thể bỏ qua
- Chia nhỏ vấn đề thành nhiều vấn đề nhỏ hơn
- Tìm cách giải quyết từng phần nhỏ
- Áp dụng các thuật toán/giải thuật phù hợp để giải quyết mỗi phần đó



Khả năng về TDLT không chỉ đơn thuần là giỏi về thuật toán!
⇒ Là kỹ năng cần được tích lũy, luyện tập thường xuyên

Để rèn luyện tư duy lập trình hiệu quả:

- Nắm chắc các kiến thức, kỹ năng lập trình cơ bản (ngôn ngữ nào? biến, hàm, câu điều kiện, vòng lặp...)
- Thực hành thói quen sử dụng thuật toán
- Tư duy kiểu sản phẩm
- Suy nghĩ như một tester
- Làm việc nhóm

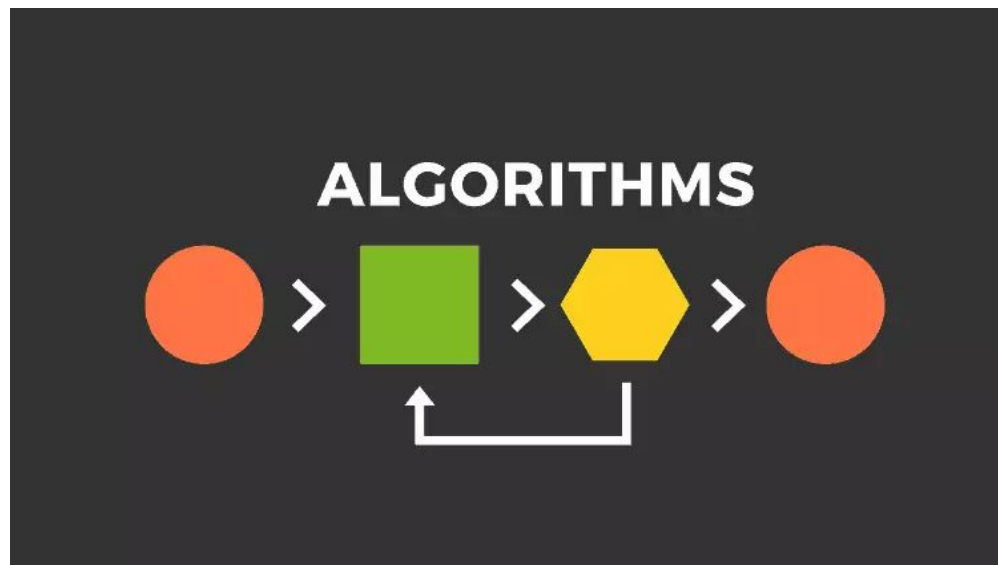


Tham khảo: Tư Duy Lập Trình (Nguyễn Thành Hiếu)
<https://www.youtube.com/watch?v=xnFcC7I-hDI>

<https://cyberlab.edu.vn/>

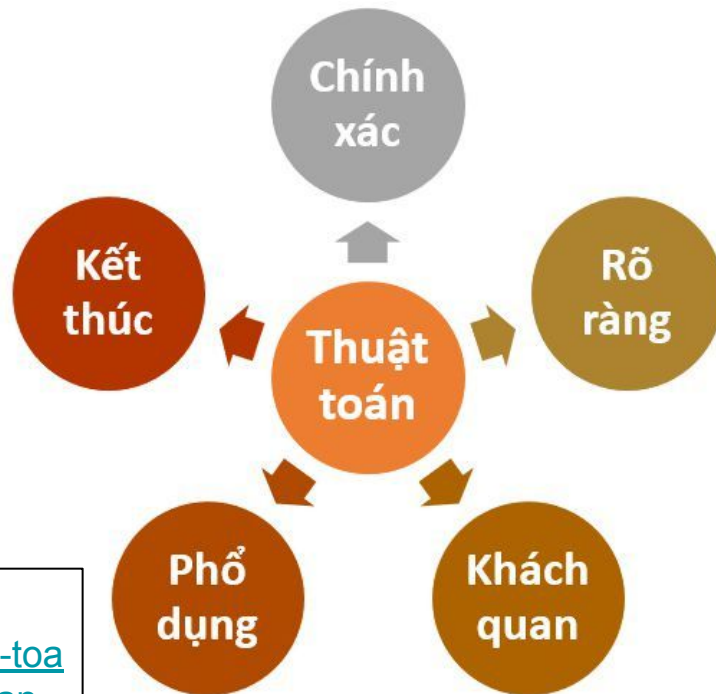
Giải thuật (algorithm): để giải quyết mỗi bài toán, ta cần tạo ra giải thuật

Là phương pháp (hiệu quả) để giải quyết một bài toán



Giải thuật (algorithm): để giải quyết mỗi bài toán, ta cần tạo ra giải thuật

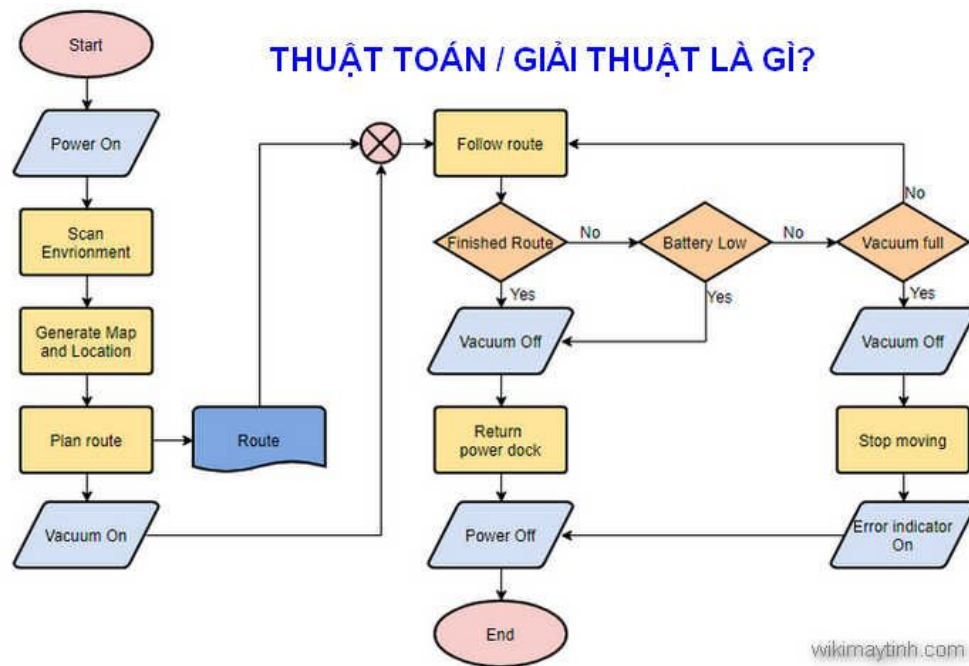
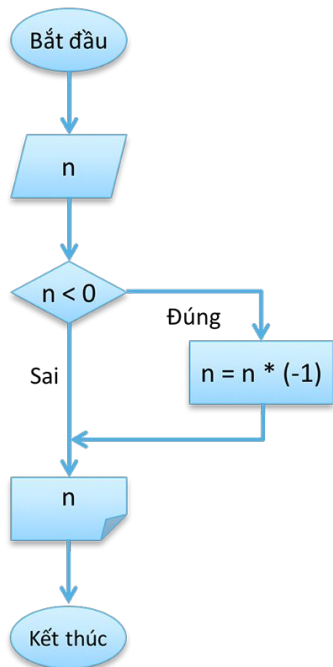
- Gồm tập hợp các thủ tục để hướng dẫn máy tính thực hiện một nhiệm vụ
- Là một phần không thể thiếu trong IT
- Thuật toán phải đảm bảo 05 tính chất



Tham khảo:







<https://gochocit.com/ky-thuat-lap-trinh/thuat-toan-la-gi-cac-phuong-phap-bieu-dien-thuat-toan>

Lược đồ giải thuật (lưu đồ thuật toán - flowchart / algorithm diagram)?



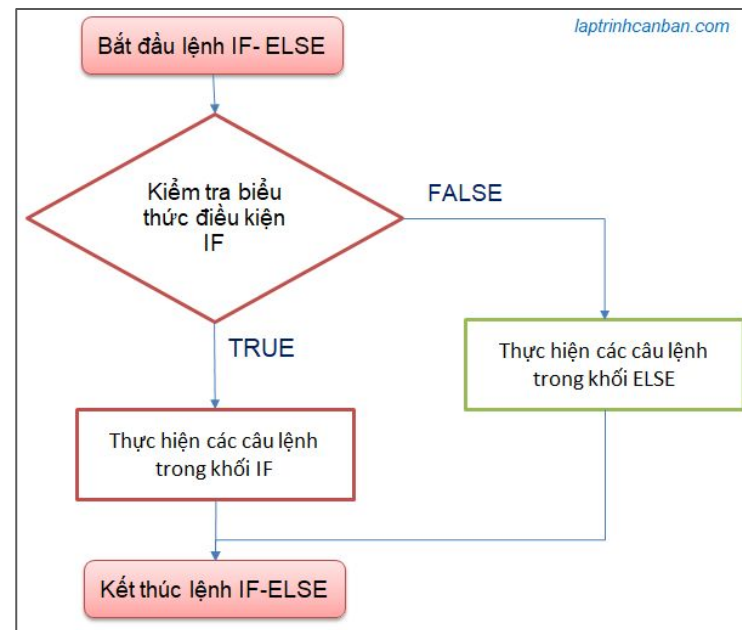
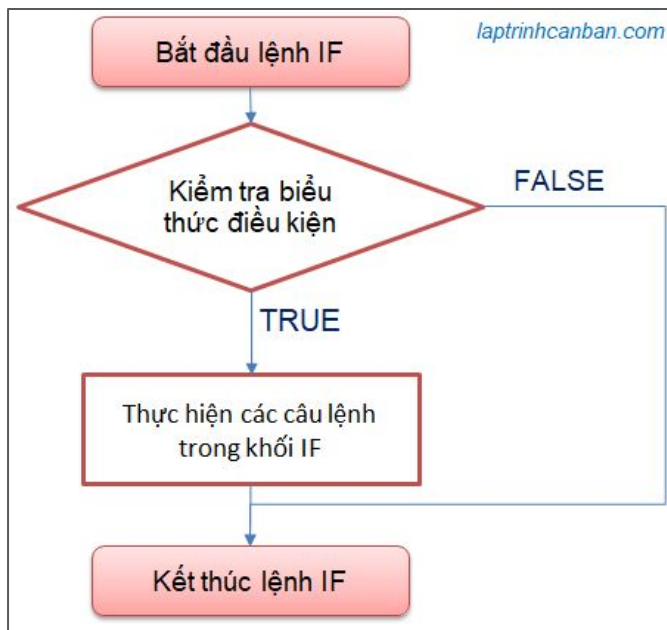
wikimaytinh.com

Lược đồ giải thuật (lưu đồ thuật toán): các ký hiệu thường dùng

Ký hiệu	Mô tả
	Điểm bắt đầu và kết thúc một thuật toán.
	Thao tác nhập hay xuất dữ liệu.
	Khối xử lý công việc.
	Khối quyết định chọn lựa.
	Dòng tính toán, thao tác của chương trình.
	Khối lệnh gọi hàm (chương trình con)

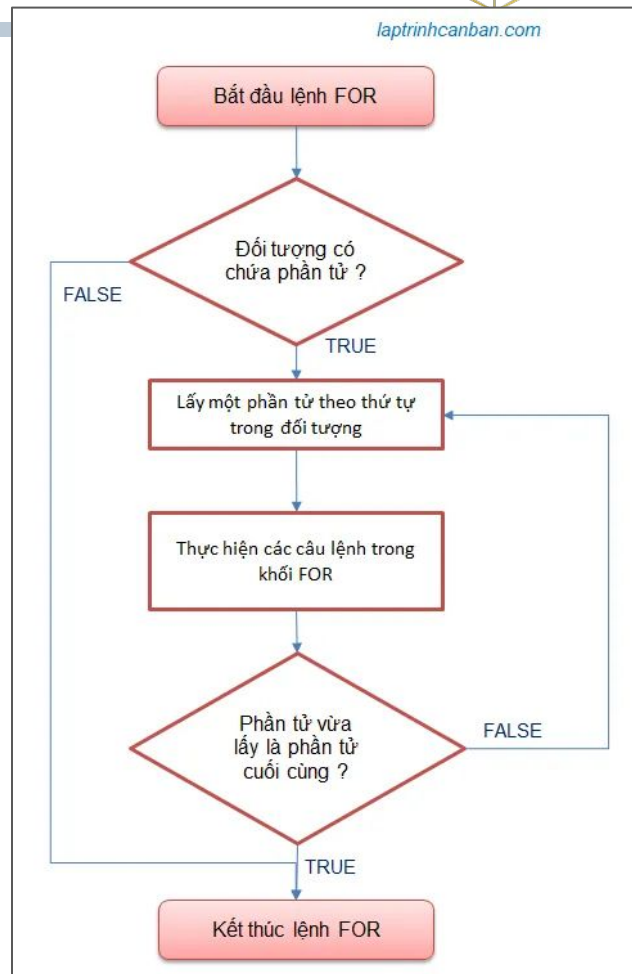
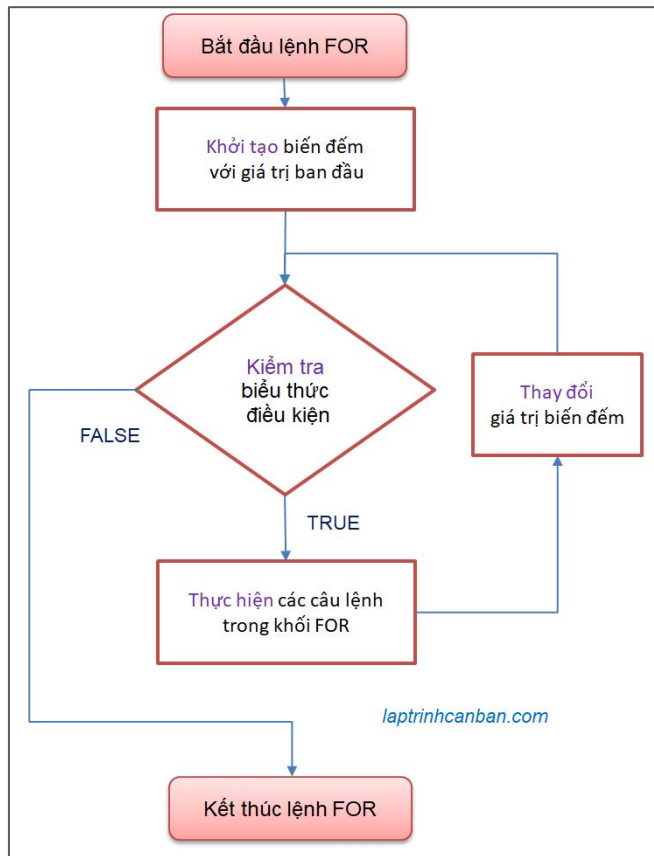
Lược đồ giải thuật:

- Câu điều kiện



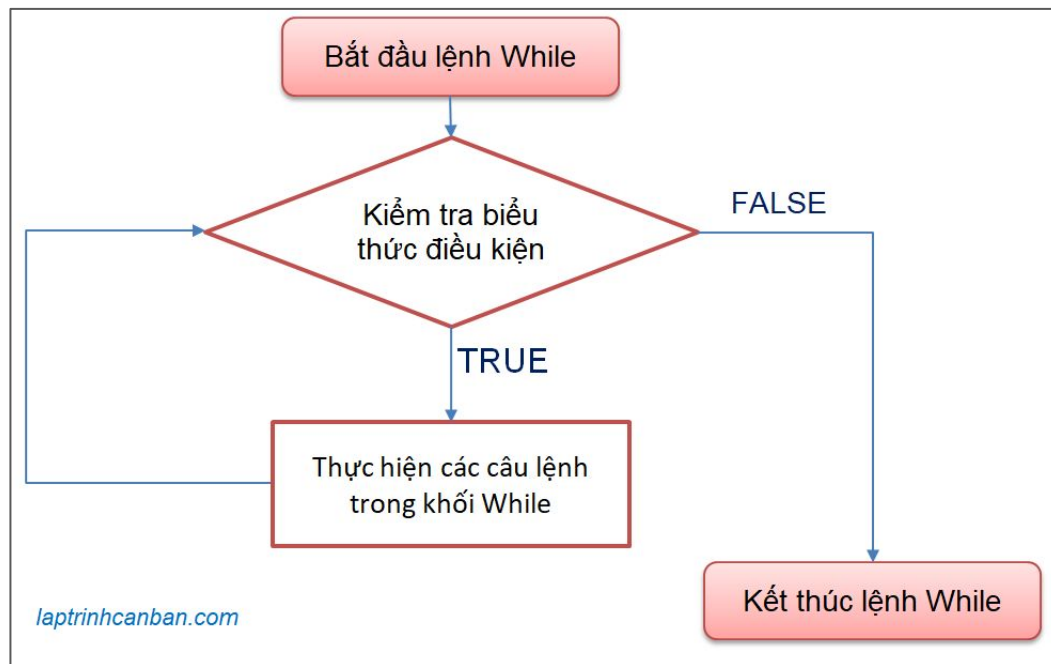
Lược đồ giải thuật:

- Vòng lặp for

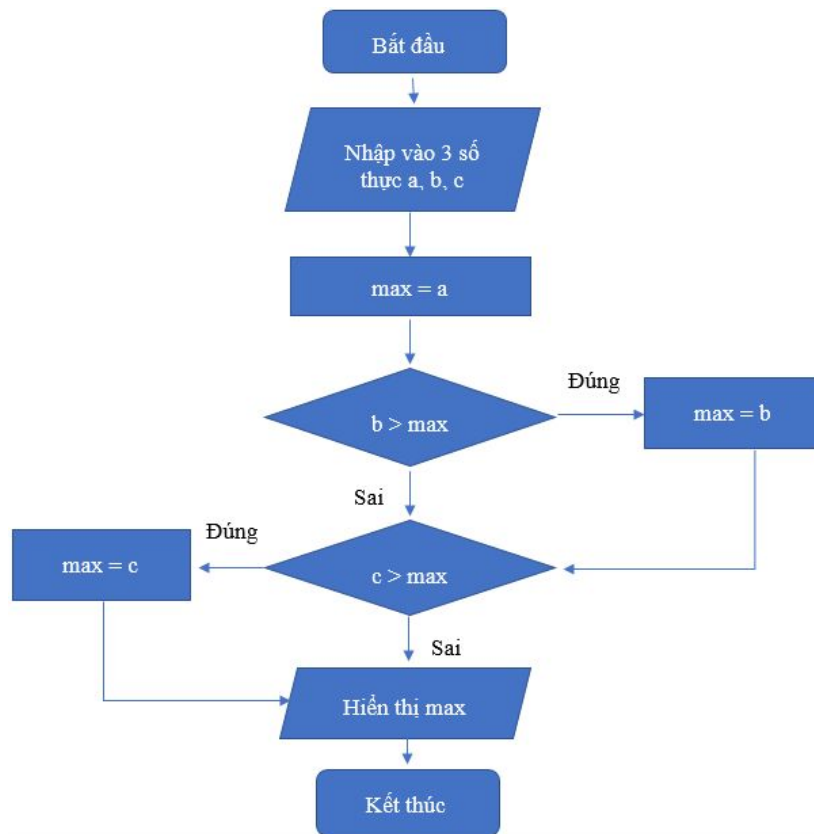


Lược đồ giải thuật:

- Vòng lặp while



Ví dụ: tìm giá trị lớn nhất trong 3 số

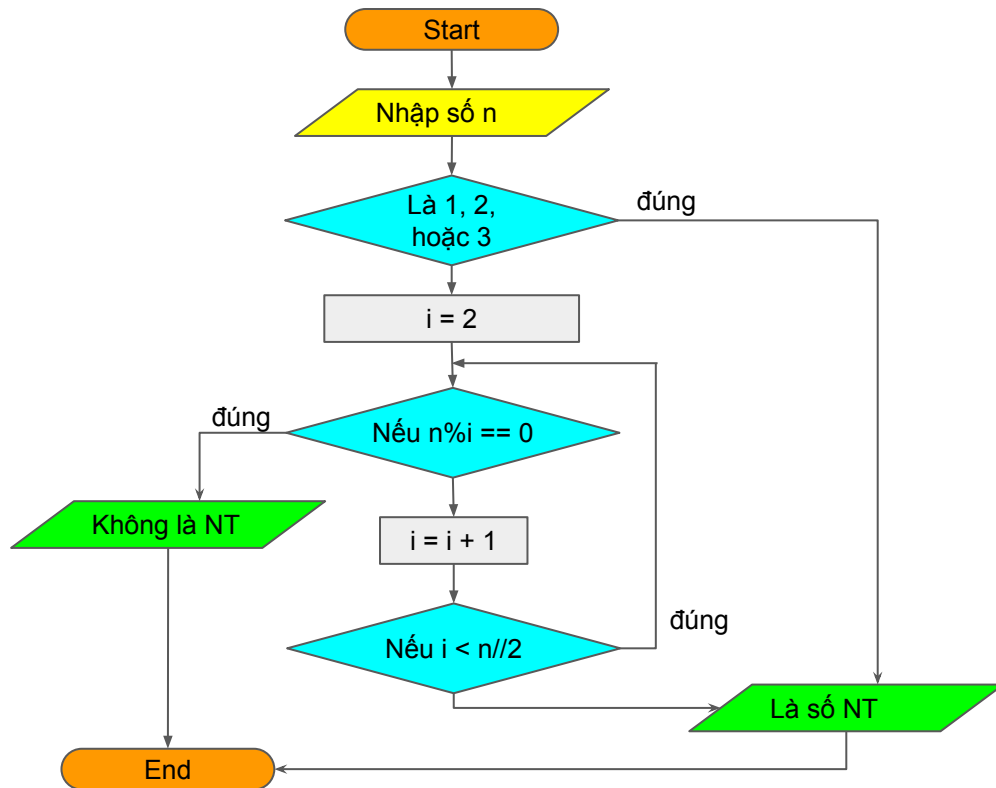


Ví dụ: xác định số nguyên n bất kỳ là số nguyên tố hay không?

- Suy nghĩ ra lời giải:
 - Các số chia hết cho 1 và chính nó (trừ các số 1, 2, 3)
⇒ Vậy chỉ cần kiểm tra các số từ 4 trở lên
 - Chỉ cần kiểm tra xem n có chia hết cho các số nhỏ hơn n
⇒ Các số từ 2, 3, 4, ... \sqrt{n}
⇒ Để đơn giản, ta tạm thời chọn lặp đến $n//2$
 - Thực hiện vòng lặp cho các số trong danh sách từ 2 đến $n//2$

Ví dụ: xác định số nguyên n bất kỳ là số nguyên tố hay không?

- Lược đồ:



Mã giả: một dạng ngôn ngữ hình thức (gần với ngôn ngữ tự nhiên) để mô tả các bước của thuật toán

⇒ Lưu ý: mã giả không thực thi được trong chương trình

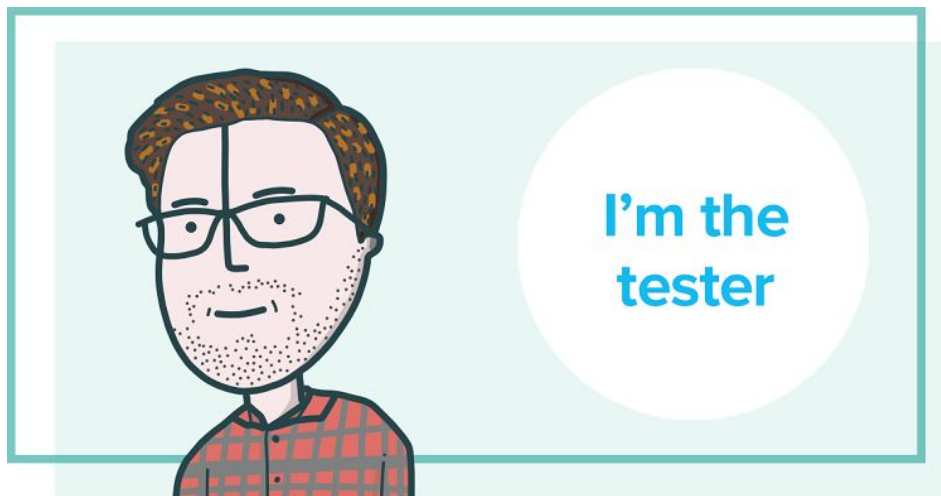
```

procedure Insert( $x, y$  : vertex;  $w_{xy}$  : weight);
1. begin {each vertex  $q$  is supposed to have  $\text{mark}(q) = \text{false}$ }
   Step 1
2.   insert edge  $(x, y)$  with weight  $w_{xy}$  in graph  $G$ ;
3.   insert  $y$  and  $x$  in  $B(x)$  and  $B(y)$ , with priorities  $d(y) - w_{xy}$  and  $d(x) - w_{xy}$ , respectively;
   Step 2
4.   if  $d(x) + w_{xy} \geq d(y)$  then EXIT; {no distance from the source has been improved}
   Step 3
5.    $Q \leftarrow \emptyset$ ; {initialization of the global heap}
6.   insert in  $Q$  vertex  $y$  with priority  $d(x) + w_{xy}$  and candidate parent  $x$ ;
   Step 4
7.   while Non.Empty( $Q$ ) do
8.     begin
9.       delete from  $Q$  the element  $q$  with minimum priority  $b_q$  and candidate parent  $z$ ;
10.       $d(q) \leftarrow b_q$ ; {update the distance of  $q$  from the source}
11.       $\text{mark}(q) \leftarrow \text{true}$ ;
12.      make  $z$  the new parent of  $q$  in  $T(s)$ ;
13.      for each possibly hot edge  $(q, r) \in B(q)$  do {i.e., if  $p_q(r) > d(q)$ }
14.        begin
15.           $t_q(r) \leftarrow d(r) - w_{q,r}$ ; {the actual value  $t_q(r) \leq p_q(r)$  is computed}
16.          if  $t_q(r) > d(q)$  and  $\text{mark}(q) = \text{false}$  {if  $(q, r)$  is a hot edge}
17.            then if  $r \notin Q$ 
18.              then insert  $r$  in  $Q$  with priority  $d(q) + w_{q,r}$  and candidate parent  $q$ 
19.            else if  $d(q) + w_{q,r} \leq \text{current-priority of } r \text{ in } Q$ 
20.              then begin
21.                update priority of  $r$  in  $Q$  to be  $d(q) + w_{q,r}$ ;
22.                set  $q$  to be the candidate parent of  $r$ ;
23.              end
24.            end
25.        end
26.      end
   Step 5
27.   for each edge  $(i, j)$  scanned in line 13 do
28.     begin
29.        $p_i(j) = d(j) - w_{i,j}$ ;
30.        $p_j(i) = d(i) - w_{i,j}$ ;
31.     end
32.   for each marked vertex  $q$  do  $\text{mark}(q) \leftarrow \text{false}$ ;
end
  
```

Phải có góc nhìn từ tester

(ở đây, có thể hiểu là người khác sử dụng CT của bạn)

**Đừng bao giờ tin tưởng 100% vào
code của mình!**



Tham khảo:

1. <https://itnavi.com.vn/blog/tu-duy-lap-trinh>
2. <https://topdev.vn/blog/tu-duy-lap-trinh-phuc-vu-cuoc-song/>
3. <https://www.youtube.com/watch?v=xnFcC7I-hDI>
4. <https://viblo.asia/p/lap-trinh-va-tu-duy-thuat-toan-sang-tao-ki-1-E375z2425GW>
5. Lập trình Python:
<https://laptrinhcanban.com/python/nhap-mon-lap-trinh-python/>

Một số mẹo để rèn luyện tư duy lập trình:

10 Ways To Improve Logic In Programming

- 1 Think to solve
- 2 Practice
- 3 Learn about data structure
- 4 Play games
- 5 Learn programming paradigms
- 6 Look at the other people's code
- 7 Take part in code challenges
- 8 Read books and solve examples
- 9 Clean code
- 10 Design Pattern

1. Vẽ lược đồ giúp kiểm tra các số nguyên tố trong 1 mảng
(Gọi tới hàm kiểm tra số nguyên tố, có lược đồ riêng)

2. Cho 1 danh sách các giá cổ phiếu theo ngày:
 - Vẽ lược đồ để kiểm tra xem giá của ngày thứ $i+1$ là tăng hay giảm so với ngày trước đó
 - Phân ra 3 trường hợp:
 - Nếu tăng trên 7%: ra lệnh bán
 - Nếu giảm trên 5%: ra lệnh mua
 - Còn lại: giữ nguyên

(Lưu ý: Thực hiện sau buổi thực hành tiếp theo)

1. Dữ liệu giá cổ phiếu: Viết vòng lặp xác định nếu giá cp lên 7% so với ngày hôm trước thì bán, xuống 5% so với ngày hôm trước thì mua

`gia_theo_ngay = [10.1, 9.5, 10.7, 10.8, 10.6, 10.0, 10.2, 9.5]`

2. Viết chương trình kiểm tra xem mỗi số trong một list có là số nguyên tố (cần viết hàm kiểm tra số NT riêng, và lưu ý trường hợp các số 1, 2, 3)
3. Viết hàm tính tiền bảo hiểm và thuế thu nhập cá nhân cho các nhân viên:
 - Input: lương, phụ cấp
 - Output: BHXH, PIT

(Lưu ý: đây là bài tập để rèn luyện lập trình, không nhất thiết phải theo đúng các quy định cụ thể của việc tính BH/thuế)

THANK YOU!

