

LẬP TRÌNH PYTHON CƠ BẢN

(Basic Python Programming)

Th.S Nguyễn Hoàng Thành

Email: thanhnh@ptithcm.edu.vn

Tel: 0909 682 711

CẤU TRÚC ĐIỀU KHIỂN

1. CÁC KHÁI NIỆM VỀ LỆNH VÀ KHỐI LỆNH
2. CẤU TRÚC ĐIỀU KHIỂN Rẽ NHÁNH, LỰA CHỌN
3. CẤU TRÚC ĐIỀU KHIỂN Rẽ NHÁNH, LẶP

1. CÁC KHÁI NIỆM VỀ LỆNH VÀ KHỎI LỆNH

1.1 Câu lệnh - statement

- Python sẽ thông dịch từng câu lệnh (statement) để thực thi. Một statement trong Python thường được viết trong 1 dòng. Chúng ta không cần thiết phải thêm dấu chấm phẩy ; vào cuối mỗi câu lệnh. Ví dụ:

```
>>> a = 5
>>> b = 10
>>> print("Tong =", a + b)
Tong = 15
```

- Chúng ta cũng có thể **đặt nhiều câu lệnh** trong một dòng bằng cách sử dụng dấu chấm phẩy ; như sau:

```
>>> a = 1; b = 2; c = 3
```

1.1 Câu lệnh – statement (tt)

- Chúng ta có thể viết một câu lệnh trên nhiều dòng bằng cách sử dụng thích hợp các ký tự như ký tự tiếp tục dòng (`\`), dấu ngoặc đơn `()`, ngoặc vuông `[]`, ngoặc nhọn `{}`.

```
>>> a = 1 + 2 + 3 + \
...     4 + 5 + 6 + \
...     7 + 8 + 9
>>> a = (1 + 2 + 3 +
...     4 + 5 + 6 +
...     7 + 8 + 9)
>>> colors = ['red',
...            'blue',
...            'green']
```

Tiếp tục dòng khi dùng `\`, `()`, `[]`

1.1 Câu lệnh – statement (tt)

- Các câu lệnh mã hóa logic lớn hơn cho hoạt động của chương trình
 - sử dụng và biểu thức trực tiếp để xử lý các đối tượng
 - luôn tồn tại trong các **mô-đun**

Python's Statements

Statements	Role	Example
Assignment	Create references	<code>a, *b = 'good', 'bad', 'ugly'</code>
Calls and another expressions	Running functions	<code>log.write("spam, ham")</code>
print calls	Printing objects	<code>print('The Killer', joke)</code>
if/elif/else	Selecting actions	<code>if "python" in text: print(text)</code>
for/else	Sequence iterations	<code>for x in mylist: print(x)</code>
while/else	General Loops	<code>while X>Y: print('hello')</code>
pass	Empty placeholder	<code>while True: pass</code>
break	Loop exit	<code>while True: break</code>

Python's Statements

Statements	Role	Example
Continue	Loop continue	<pre>while True: if skiptest():continue</pre>
def	Functions and methods	<pre>def f(a, b, c=1, *d): print(a + b + c + d[0])</pre>
return	Function results	<pre>def f(a, b, c=1, *d): return a + b + c + d[0]</pre>
yield	Generator functions	<pre>def gen(n): for i in n: yield i*2</pre>
global	Namespaces	<pre>x = 'old' def function(): global x, y; x = 'new'</pre>
nonlocal	Namespace (3.0+)	<pre>def outer(): nonlocal x; x = 'new'</pre>

Python's Statements

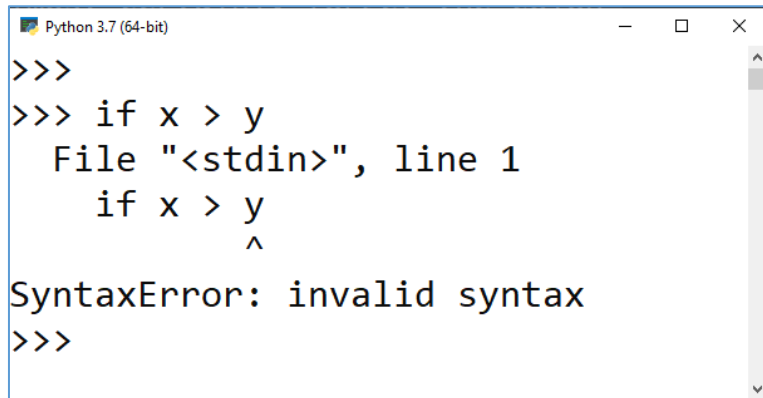
Statements	Role	Example
import	Module access	import sys
from	Attribute access	from sys import stdin
class	Building objects	class Subclass(Superclass): staticData = [] def method(self):pass
Try/except/finally	Generator functions	try: action() except: print('action error')
raise	Triggering exceptions	raise EndSearch(location)
assert	Debugging checks	assert X > Y, 'X too small'
with/as	Context manager (2.6+)	with open('data') as myfile: process(myfile)

Python's Statements

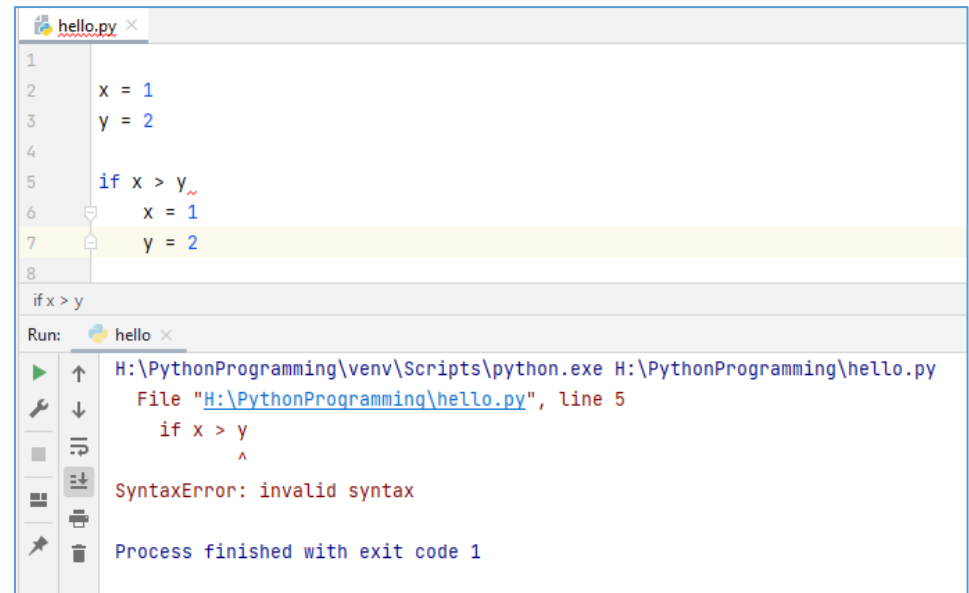
Statements	Role	Example
del	Deleting references	del data[k] del data[i:j] del obj.attr del variable

1.2 Khối lệnh – code block

- Thành phần cú pháp mới trong Python là ký tự dấu hai chấm (:)
- Tất cả các câu lệnh ghép trong Python đều tuân theo cùng một mẫu chung của dòng tiêu đề được kết thúc bằng dấu hai chấm:
 - Dòng tiêu đề:
 - Khối câu lệnh lồng nhau



```
>>>
>>> if x > y
    File "<stdin>", line 1
        if x > y
            ^
SyntaxError: invalid syntax
>>>
```



```
hello.py x
1
2   x = 1
3   y = 2
4
5   if x > y
6       x = 1
7       y = 2
8
if x > y
Run: hello x
H:\PythonProgramming\venv\Scripts\python.exe H:\PythonProgramming\hello.py
File "H:\PythonProgramming\hello.py", line 5
    if x > y
        ^
SyntaxError: invalid syntax
Process finished with exit code 1
```

1.2 Khối lệnh – code block (tt)

- Cuối dòng là kết thúc câu lệnh
- Kết thúc thật đầu dòng là kết thúc khối

```
Python 3.7 (64-bit)
>>>
>>> if x < y:
...     x = 1;
...     y = 2;
...
>>>
```

```
Python 3.7 (64-bit)
>>>
>>> if x < y:
...     x = 1
...     y = 2
...
>>>
```

```
Python 3.7 (64-bit)
>>>
>>> if x < y:
...     x = 1; y = 2
...
>>>
```

Tại sao dùng cú pháp thụt đầu dòng?

- Python gần như buộc các lập trình viên phải tạo ra mã thống nhất, thông thường và dễ đọc.
 - phải sắp xếp mã của bạn:
 - theo chiều dọc,
 - trong cột,
 - theo cấu trúc logic của nó.

2. CẤU TRÚC ĐIỀU KHIỂN RẼ NHÁNH

2.1 Đặt vấn đề

❖ **Bài toán:** Xây dựng chương trình máy tính thực hiện các yêu cầu sau:

- Nhập vào tên và điểm thi của một sinh viên
- Hiển thị thông tin về kết quả thí sinh trượt hoặc đỗ biết rằng:
 - ☐ Thí sinh trượt nếu điểm thi **Nhỏ hơn 5**
 - ☐ Thí sinh đỗ nếu điểm thi **Lớn hơn hoặc bằng 5**

Xác định yêu cầu bài toán

- ❖ **Nhập dữ liệu** đầu vào là 2 biến:
 - ❖ **tên** (name) kiểu chuỗi
 - ❖ **điểm** (mark) kiểu thực
- ❖ **Hiển thị** biến name **ra màn hình** và **kết quả** thí sinh trượt / đỗ

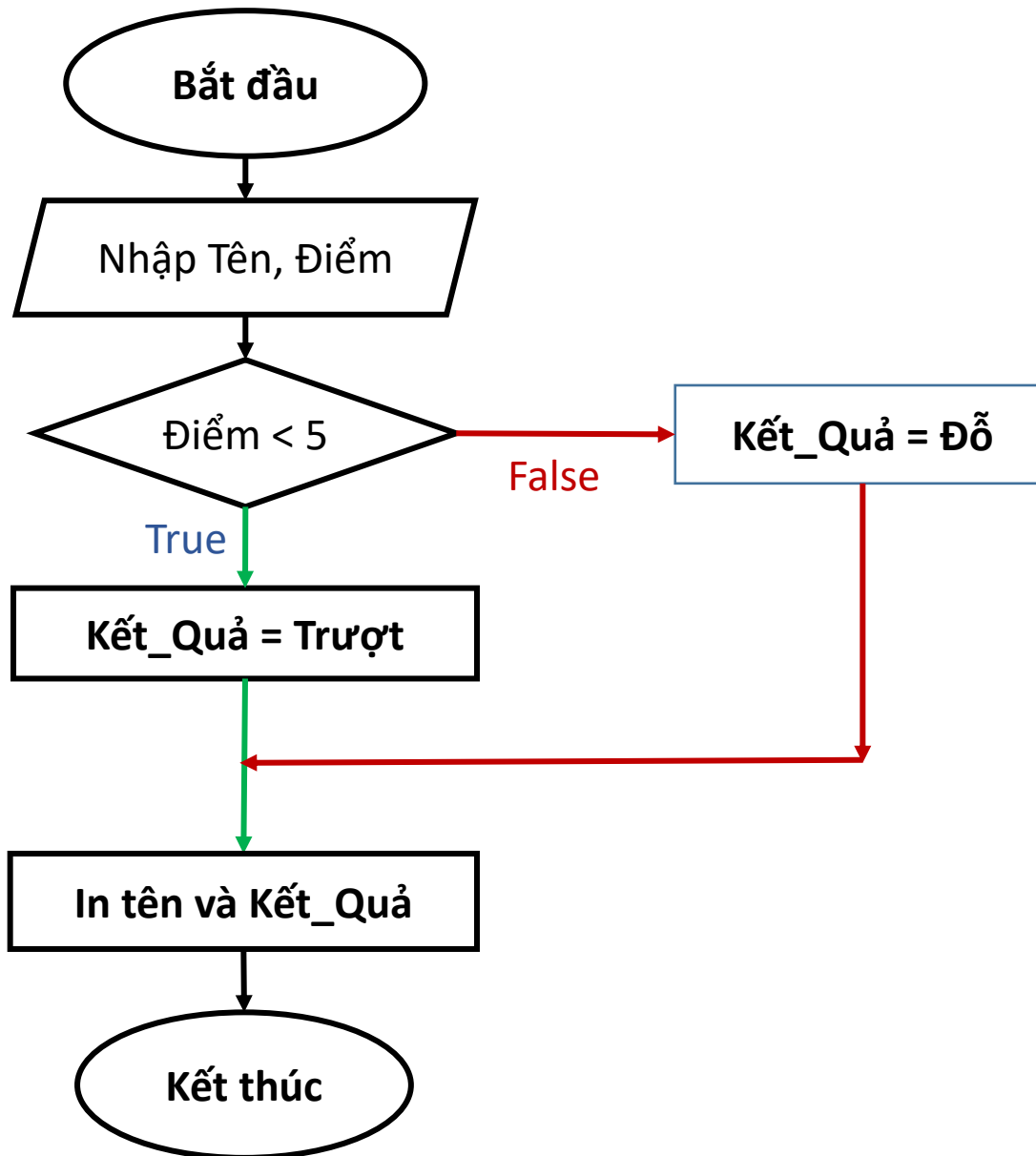
Cách giải quyết

- Sử dụng cấu trúc rẽ nhánh để xét nếu điểm mark lớn hơn hoặc bằng 5 thì đỗ, ngược lại sẽ bị trượt.

Thuật toán

- Input: Nhập tên, Nhập điểm
- Xử lý:
 - Nếu Điểm Nhỏ hơn 5, kết quả là Trượt
 - Nếu Điểm Lớn hơn hoặc bằng 5, kết quả là Đủ
- Output:
 - In ra Tên thí sinh và Kết quả ra màn hình

Lưu đồ thuật toán



Viết chương trình

Chú thích, comment

ket_qua_thi.py

```
name = input('Nhập tên: ') #Nhập giá trị biến chuỗi name
mark = float(input('Nhập điểm: ')) #Nhập giá trị biến thực mark
if mark < 5:
    ketQua = "Trượt"
else:
    ketQua = "Đỗ"
print(name, " - ", ketQua)
```

Viết chương trình

ket_qua_thi.py

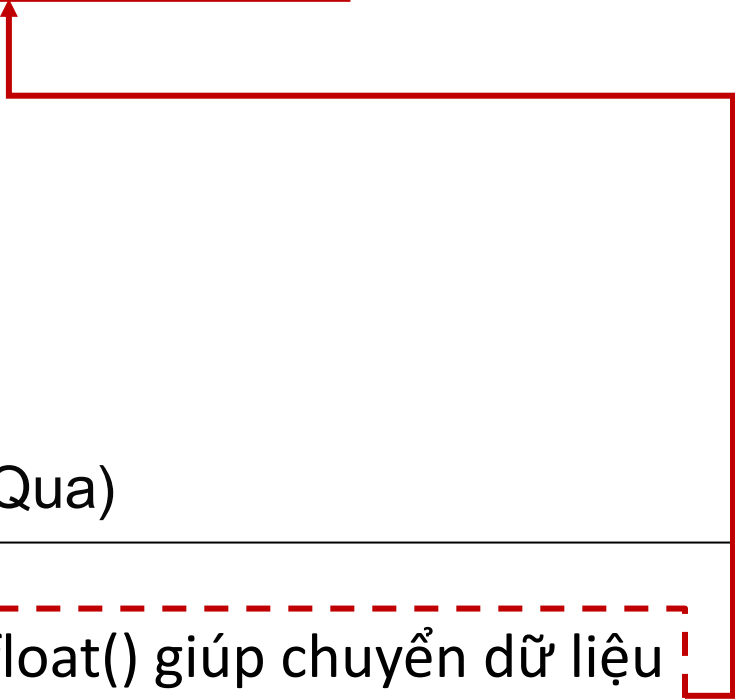
```
name = input('Nhập tên: ') #Nhập giá trị biến chuỗi name
mark = float(input('Nhập điểm: ')) #Nhập giá trị biến thực mark
if mark < 5:
    ketQua = "Trượt"
else:
    ketQua = "Đỗ"
print(name, " - ", ketQua)
```

Hàm input() giúp nhận dữ liệu
kiểu chuỗi từ bàn phím

Viết chương trình

ket_qua_thi.py

```
name = input('Nhập tên: ')    #Nhập giá trị biến chuỗi name
mark = float(input('Nhập điểm: ')) #Nhập giá trị biến thực mark
if mark < 5:
    ketQua = "Trượt"
else:
    ketQua = "Đỗ"
print(name, " - ",ketQua)
```

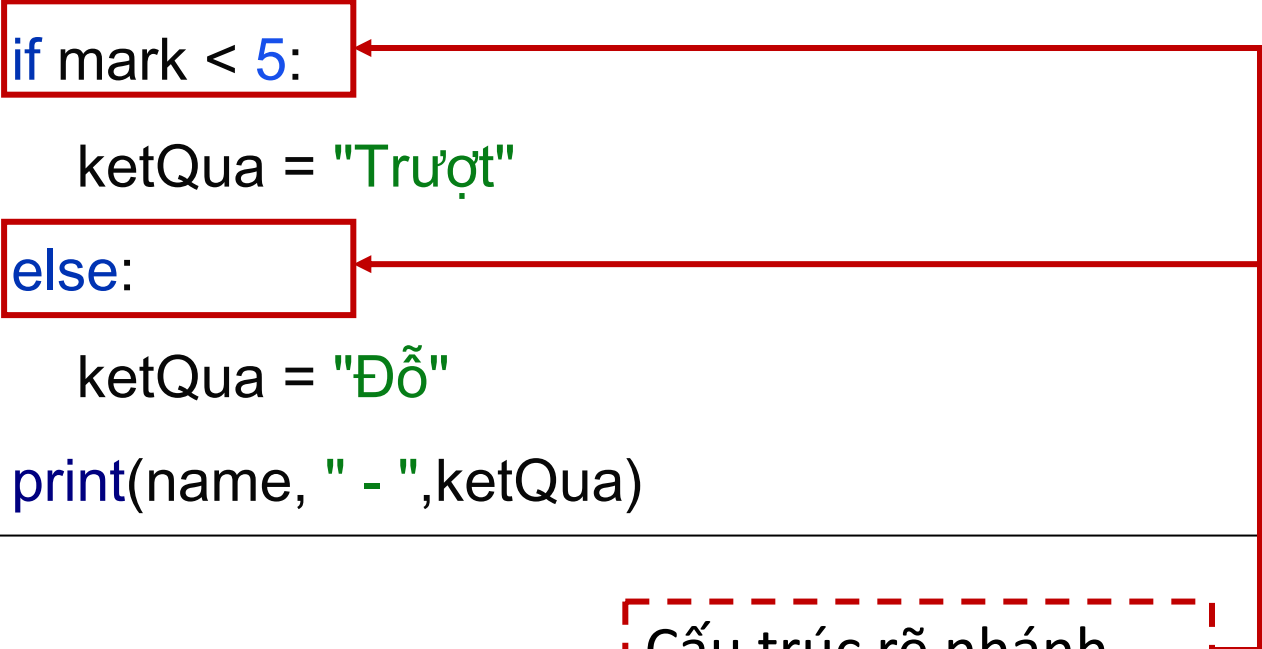


Hàm float() giúp chuyển dữ liệu
kiểu chuỗi về số thực

Viết chương trình

ket_qua_thi.py

```
name = input('Nhập tên: ')    #Nhập giá trị biến chuỗi name
mark = float(input('Nhập điểm: ')) #Nhập giá trị biến thực mark
if mark < 5:
    ketQua = "Trượt"
else:
    ketQua = "Đỗ"
print(name, " - ",ketQua)
```



Cấu trúc rẽ nhánh

2.2 Biểu thức logic

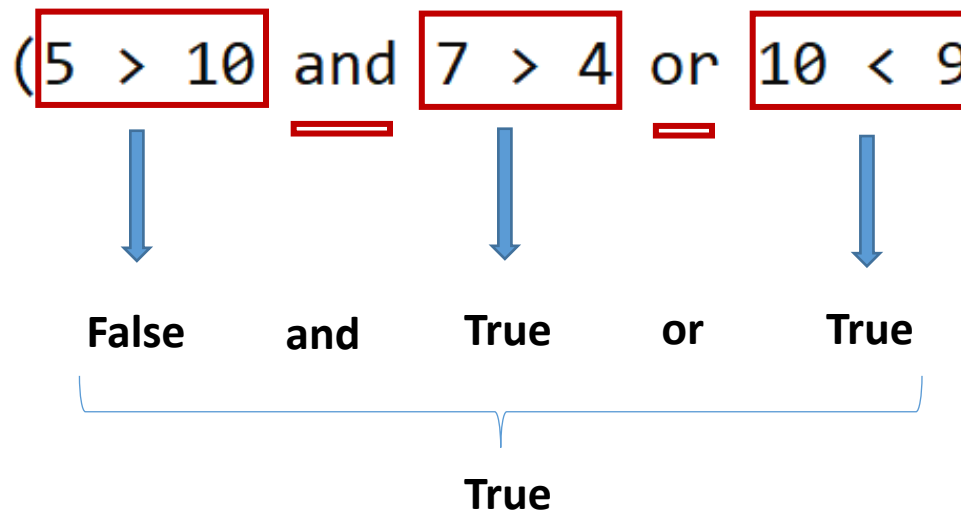
- Một biểu thức logic luôn cho kết quả trả về là một trong hai giá trị đúng (True) hoặc sai (False)

```
>>> kt = (5 > 10)
>>> kt
False
```



5 nhỏ 10 là biểu thức sai

```
>>> kt = (5 > 10) and 7 > 4 or 10 < 90
>>> kt
True
```



False and True or True

True

2.3 Cấu trúc rẽ nhánh với if

- if là từ chỉ sự bắt đầu của một cấu trúc câu điều kiện trong tiếng Anh, nghĩa tiếng Việt là “nếu”, “giả mà”,...
- Ví dụ:
 - Nếu tôi thi đỗ đại học thì tôi sẽ chọn học ngành CNTT.
 - Nếu hôm nay trời mưa, tôi sẽ ở nhà
 - Nếu m lớn hơn 0 thì m là số dương

2.3.1 Cấu trúc if

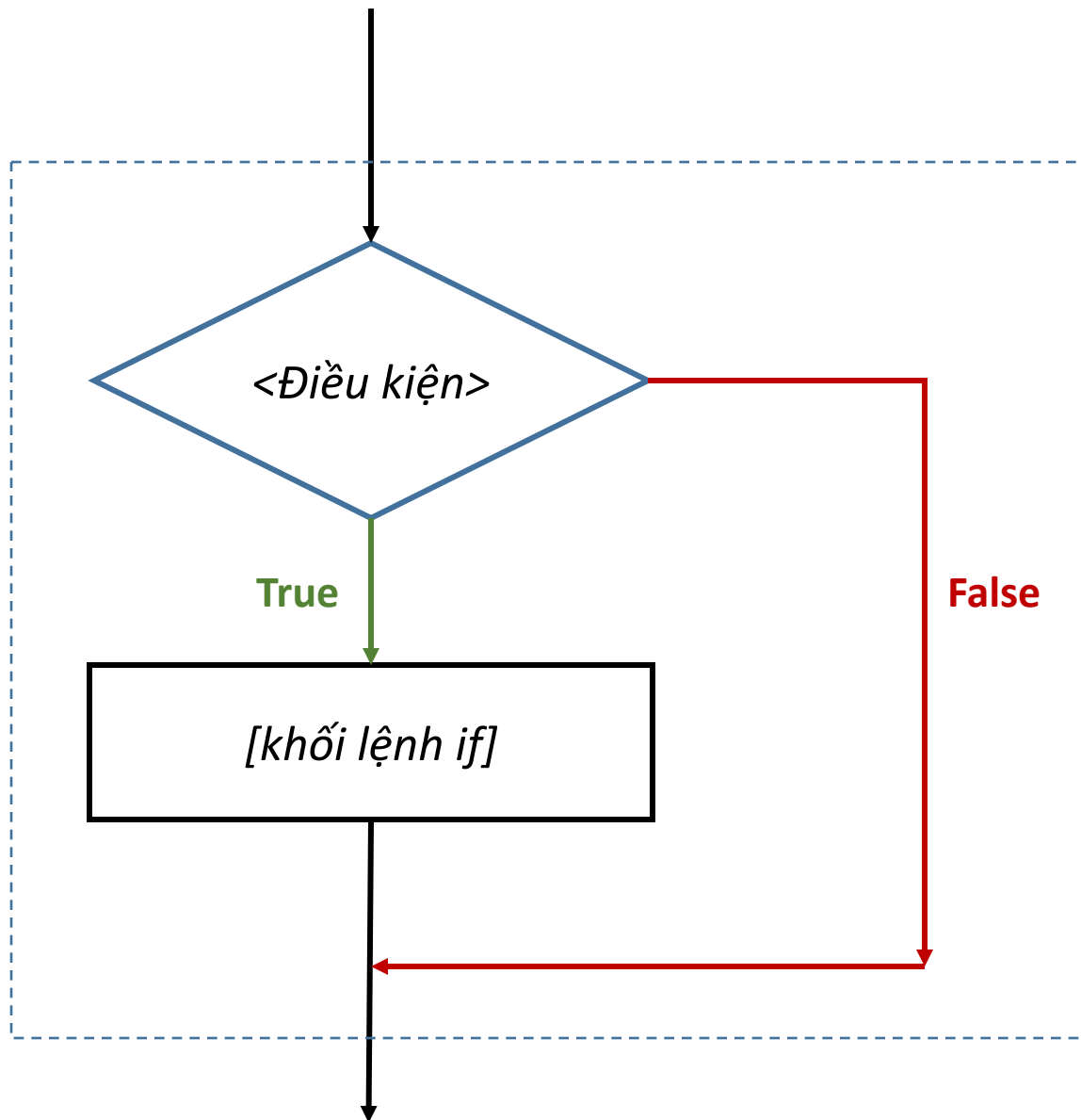
if <điều kiện>:

[khối lệnh if]

Trong đó:

- **<điều kiện>**: Luôn cho kết quả trả về là một trong hai giá trị đúng (True) hoặc sai (False)
- **[Khối lệnh if]**: Bao gồm một hoặc nhiều lệnh được chạy khi biểu thức logic có giá trị đúng (True), các câu lệnh trong khối lệnh có lẽ trái thụt dòng so với câu lệnh if

Lưu đồ thuật toán cấu trúc if



Ví dụ 1:

- Nhập số nguyên a từ bàn phím. Nếu a lớn hơn 5 thì gán b bằng a

```
>>> a = int(input("Nhập a: "))
Nhập a: 7
>>> if a > 5:
...     b = a
...
>>> print("a:{0}, b:{1}".format(a, b))
a:7, b:7
```

Dùng hàm input
nhập từ bàn phím

Dùng hàm int()
để đưa chuỗi về
số nguyên

Ví dụ 1:

- Nhập số nguyên a từ bàn phím. Nếu a lớn hơn 5 thì gán b bằng a

```
>>> a = int(input("Nhập a: "))
Nhập a: 7
>>> if a > 5:
...     b = a
...
>>> print("a:{0}, b:{1}".format(a, b))
a:7, b:7
```

Lệnh if với điều kiện $a > 5$

Khối lệnh sẽ thực thi nếu điều kiện là True

Ví dụ 2:

- Nhập số nguyên x từ bàn phím. Kiểm tra nếu $x > 0$ và nhỏ hơn 50. Nếu thỏa thì hiển thị x ra màn hình.

```
x = int(input("Nhập x:"))  
  
if x > 0 and x < 50:  
    print(x, ": Là số dương nhỏ hơn 50")
```

```
Nhập x:10  
10 : Là số dương nhỏ hơn 50
```

2.3.2 Cấu trúc if else

if <điều kiện>:

[khối lệnh if]

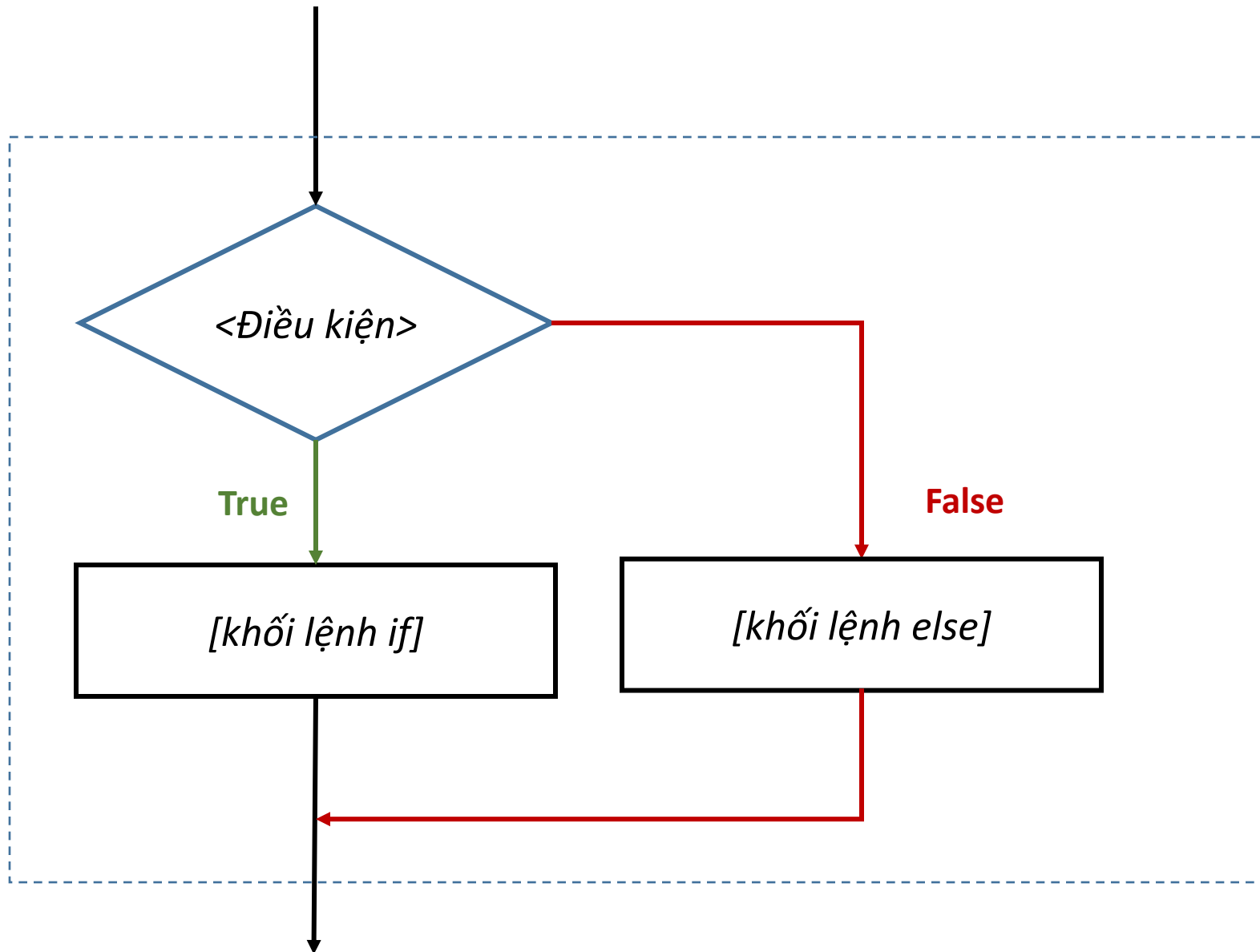
else:

[khối lệnh else]

Trong đó:

- **<điều kiện>**: Luôn cho kết quả trả về là một trong hai giá trị đúng (True) hoặc sai (False)
- **[Khối lệnh if]**: Bao gồm một hoặc nhiều lệnh được chạy khi biểu thức logic có giá trị đúng (True), các câu lệnh trong khối lệnh có lẽ trái thụt dòng so với câu lệnh if
- **[Khối lệnh else]**: Là khối lệnh được chạy khi biểu thức logic khi biểu thức logic có giá trị sai (False), các câu lệnh trong khối được thụt dòng sang phải so với câu lệnh else.

Lưu đồ thuật toán cấu trúc if else



Ví dụ 3:

- Nhập số nguyên num từ bàn phím. Kiểm tra nếu num > hoặc bằng 0 và nhỏ hơn hoặc bằng 20. Nếu thỏa thì hiển thị “Thank you” ra màn hình, nếu không thì hiển thị “Out of range” ra màn hình

```
num = int(input("Enter a number between 10 and 20: "))
if num >= 10 and num <= 20:
    print("Thank you")
else:
    print("Out of range")
```

```
Enter a number between 10 and 20: 15
Thank you
```

Ví dụ 4:

- Nhập số nguyên num từ bàn phím. Kiểm tra num là số chẵn nằm trong $[1, 5]$ thì hiển thị “Thank you”, nếu không thì hiển thị “Incorrect”

```
num = int(input("Enter an EVEN number between 1 and 5: "))
if num == 2 or num == 4:
    print("Thank you")
else:
    print("Incorrect")
```

```
Enter an EVEN number between 1 and 5: 2
Thank you
```

2.3.3 Cấu trúc if-elif-else:

if <điều kiện 1>:

[khối lệnh 1]

elif <điều kiện 2>:

[khối lệnh 2]

...

elif <điều kiện n>:

[khối lệnh n]

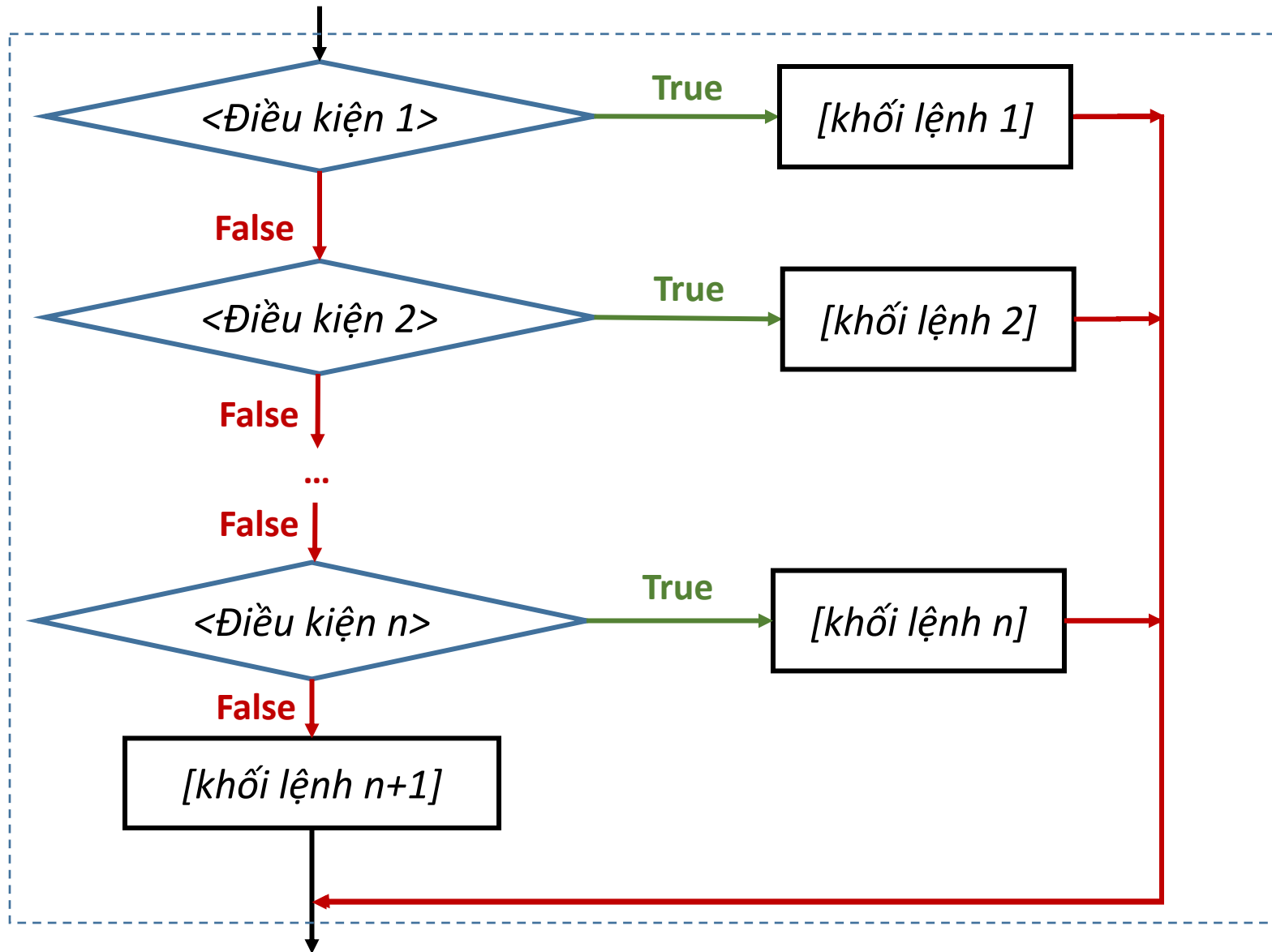
else:

[khối lệnh n+1]

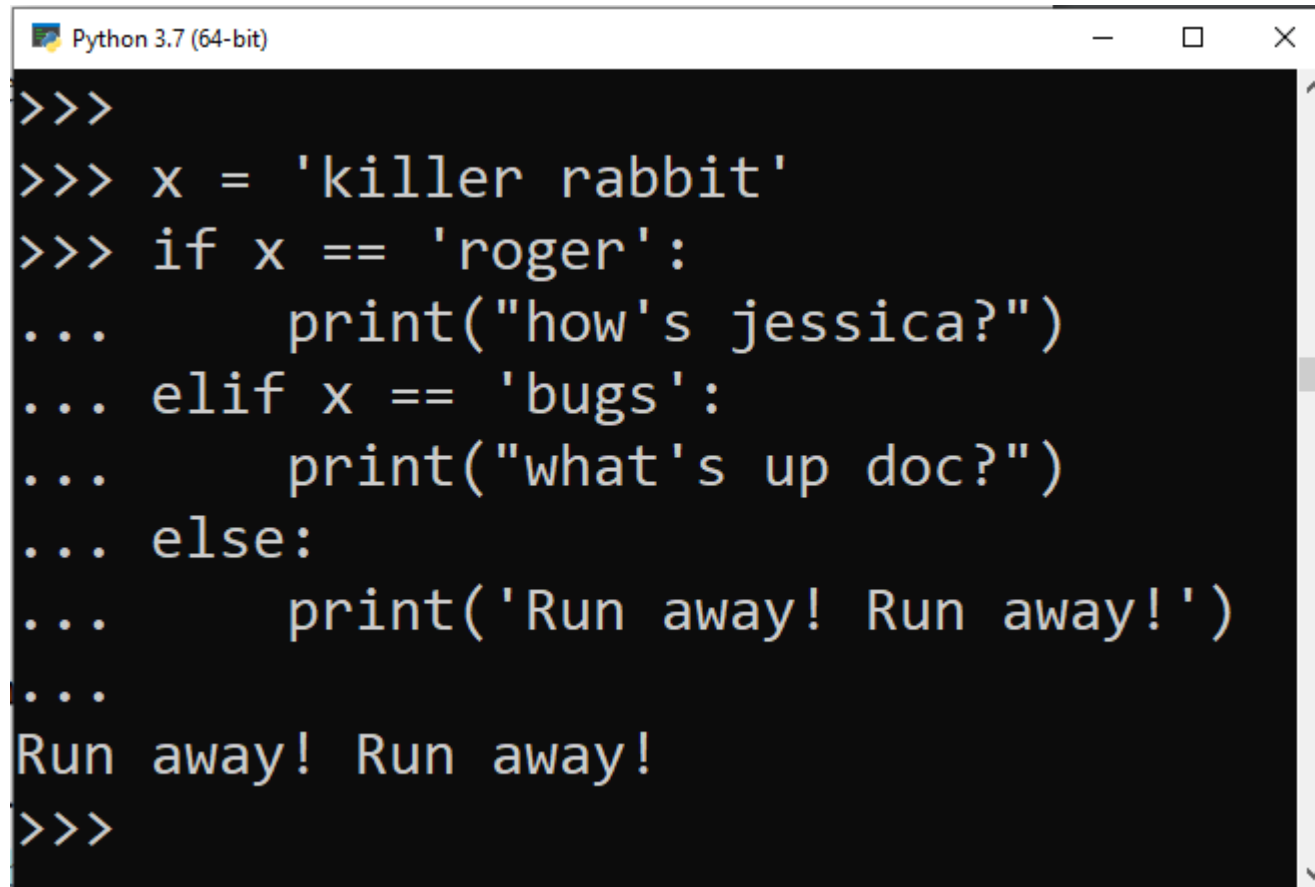
Trong đó:

- **<điều kiện 1, 2,..., n>**: Là các biểu thức logic chỉ điều kiện cho kết quả đúng (True) hoặc sai (False)
- **[Khối lệnh 1, 2,..., n]**: Là khối lệnh được chạy khi biểu thức logic tương ứng 1, 2,..., n có giá trị đúng (True), các câu lệnh trong khối được thực dòng sang phải so với câu lệnh if, elif.
- **[Khối lệnh n + 1]**: Là khối lệnh được chạy khi biểu thức logic khi tất cả các biểu thức logic từ 1 đến n có giá trị sai (False), các câu lệnh trong khối được thực dòng sang phải so với câu lệnh else.

Lưu đồ thuật toán cấu trúc if-elif-else:

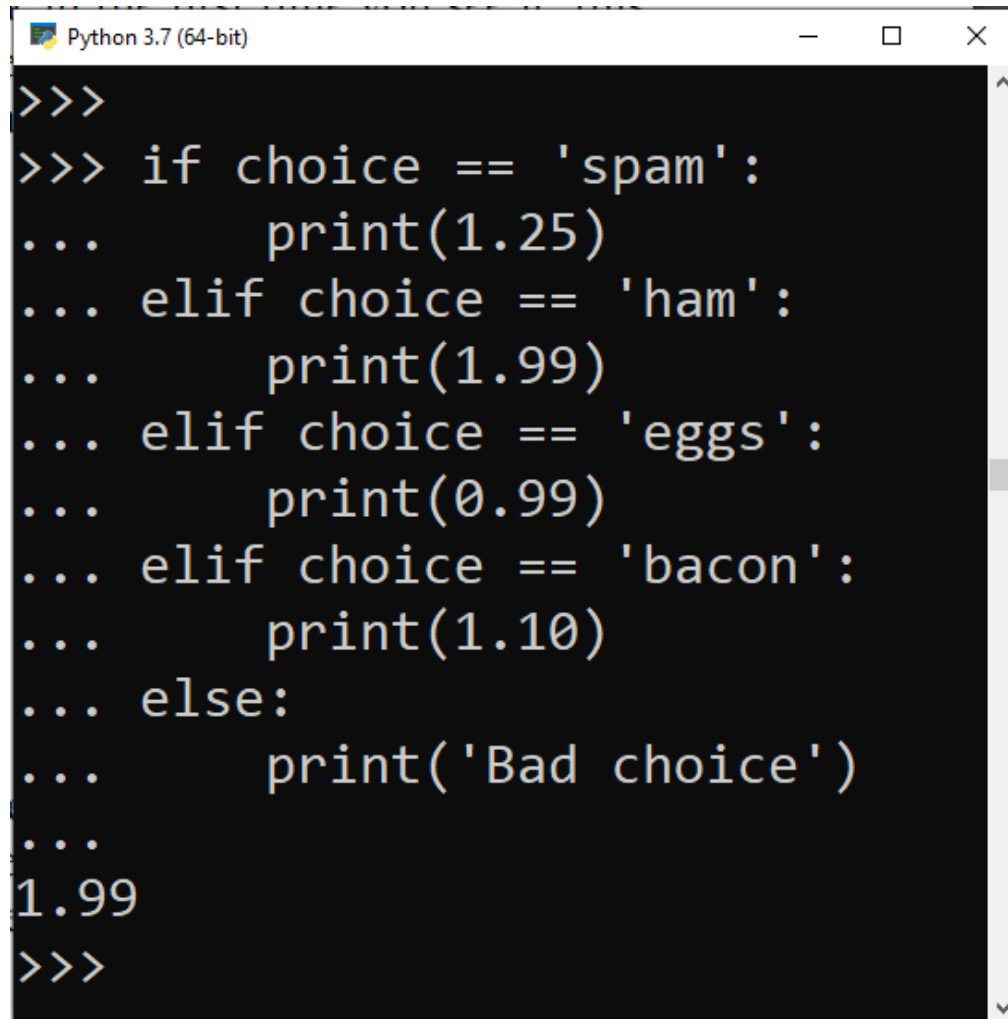


Ví dụ 5:



```
Python 3.7 (64-bit)
>>>
>>> x = 'killer rabbit'
>>> if x == 'roger':
...     print("how's jessica?")
... elif x == 'bugs':
...     print("what's up doc?")
... else:
...     print('Run away! Run away!')
...
Run away! Run away!
>>>
```

Ví dụ 6:

A screenshot of a Python 3.7 (64-bit) window. The window has a title bar with the Python logo and text "Python 3.7 (64-bit)", and standard window controls (minimize, maximize, close). The main area is a black terminal with yellow text. It shows a series of prompts and code. The first prompt is ">>>". The second prompt is ">>> if choice == 'spam':". This is followed by four lines of code, each preceded by three dots "...": "print(1.25)", "elif choice == 'ham':", "print(1.99)", "elif choice == 'eggs':", "print(0.99)", "elif choice == 'bacon':", "print(1.10)", "else:", "print('Bad choice')". This is followed by three more dots "...". The next line is the output "1.99". The final prompt is ">>>".

```
Python 3.7 (64-bit)
>>>
>>> if choice == 'spam':
...     print(1.25)
... elif choice == 'ham':
...     print(1.99)
... elif choice == 'eggs':
...     print(0.99)
... elif choice == 'bacon':
...     print(1.10)
... else:
...     print('Bad choice')
...
1.99
>>>
```

Ví dụ 8:

- Viết chương trình nhập vào 3 số thực a, b, c . Kiểm tra 3 số thực đó có thỏa mãn là độ dài 3 cạnh của 1 tam giác hay không. Nếu có, hãy tính diện tích và chu vi tam giác đó.

Ví dụ 6:

- Viết chương trình nhập vào 2 số nguyên dương là tháng và năm. Hãy tính và hiển thị số ngày của tháng trong năm đó.
- Biết rằng
 - Tháng 01, 03, 05, 07, 08, 10, 12 có 31 ngày
 - Tháng 04, 06, 09, 11 có 30 ngày
 - Tháng 02 có 28 hoặc 29 ngày tùy theo năm đó có phải là năm nhuận hay không?
 - Năm nhuận là năm chia hết cho 4.

Bài tập

- **Bài 1:** Nhập vào hai số từ bàn phím. Nếu số thứ nhất lớn hơn số thứ hai, hãy hiển thị số thứ hai trước rồi đến số thứ nhất, nếu không thì hiển thị số thứ nhất trước rồi mới đến số thứ hai.
- **Bài 2:** Yêu cầu người dùng nhập số dưới 20. Nếu họ nhập số từ 20 trở lên, hiển thị thông báo “Quá cao”, nếu không hiển thị “Cảm ơn”.
- **Bài 3:** Yêu cầu người dùng nhập một số từ 10 đến 20 (bao gồm cả 10 và 20). Nếu họ nhập số trong phạm vi này thì hiển thị thông báo “Cảm ơn”, ngược lại hiển thị thông báo “Trả lời sai”.

Bài tập (tt)

- **Bài 4:** Yêu cầu người dùng nhập màu yêu thích của họ. Nếu họ nhập “red”, “Red” hoặc “RED” hiển thị thông báo “I like red too”, ngược lại hiển thị thông báo “I don’t like [color], I prefer red”.
- **Bài 5:** Hỏi người dùng xem trời có mưa không và chuyển câu trả lời của họ thành **chữ thường**. Nếu họ trả lời “yes”, hãy hỏi xem trời có gió không. Nếu họ trả lời “yes” cho câu hỏi thứ hai này, hãy hiển thị câu trả lời “Trời quá gió để mang ô”, nếu không thì hiển thị thông báo “Take an umbrella”. Nếu họ không trả lời có cho câu hỏi đầu tiên, hãy hiển thị câu trả lời “Enjoy your day”.
- **Bài 6:** Yêu cầu người dùng nhập một số. Nếu dưới 10 thì hiển thị thông báo “Too low”, nếu từ 10 đến 20 thì hiển thị “Correct”, ngược lại hiển thị “Too high”.

Bài tập (tt)

- **Bài 7:** Yêu cầu người dùng nhập 1, 2 hoặc 3. Nếu họ nhập 1, hiển thị thông báo “Cảm ơn”, nếu họ nhập 2, hiển thị “Hoàn thành tốt”, nếu họ nhập 3, hiển thị “Đúng”. Nếu họ nhập bất cứ điều gì khác, hiển thị "Thông báo lỗi".

3. CẤU TRÚC ĐIỀU KHIỂN LẶP

3.1 Đặt vấn đề

Xét 2 bài toán là: Xây dựng chương trình hiển thị bảng nhân i và xây dựng chương trình hiển thị bảng cửu chương.

❖ **Bài toán 3.1:** Xây dựng chương trình máy tính để hiển thị bảng nhân i (giả sử chọn $i = 2$).

❑ **Phương án 1:** Dùng lệnh *print* để hiển thị bảng nhân i

❑ **Phương án 2:** Sử dụng cấu trúc lặp *for* để xây dựng chương trình giải bài toán này.

BT1. Phương án 1: Sử dụng lệnh *print*

Chương trình BT3.1.1

```
i=2
print("{0} * {1} = {2}".format(i, 1, i * 1))
print("{0} * {1} = {2}".format(i, 2, i * 2))
print("{0} * {1} = {2}".format(i, 3, i * 3))
print("{0} * {1} = {2}".format(i, 4, i * 4))
print("{0} * {1} = {2}".format(i, 5, i * 5))
print("{0} * {1} = {2}".format(i, 6, i * 6))
print("{0} * {1} = {2}".format(i, 7, i * 7))
print("{0} * {1} = {2}".format(i, 8, i * 8))
print("{0} * {1} = {2}".format(i, 9, i * 9))
print("{0} * {1} = {2}".format(i, 10, i * 10))
```

Kết quả

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

BT1. Phương án 2: Sử dụng cấu trúc *for*

Chương trình BT3.1.2

```
i=2  
for x in range(1,11):  
    print("{0} * {1} = {2}".format(i, x, i * x))
```

Kết quả

```
2 * 1 = 2  
2 * 2 = 4  
2 * 3 = 6  
2 * 4 = 8  
2 * 5 = 10  
2 * 6 = 12  
2 * 7 = 14  
2 * 8 = 16  
2 * 9 = 18  
2 * 10 = 20
```


Nhận xét

- Cả hai chương trình đều giải quyết được bài toán 1.
- Chương trình BT1.2 ngắn gọn hơn BT1.1

BT2. Phương án 1: Sử dụng lệnh *print*

Chương trình BT3.2.1

```
i=1
print("{0} * {1} = {2}".format(i, 1, i * 1))
print("{0} * {1} = {2}".format(i, 2, i * 2))
print("{0} * {1} = {2}".format(i, 3, i * 3))
print("{0} * {1} = {2}".format(i, 4, i * 4))
print("{0} * {1} = {2}".format(i, 5, i * 5))
print("{0} * {1} = {2}".format(i, 6, i * 6))
print("{0} * {1} = {2}".format(i, 7, i * 7))
print("{0} * {1} = {2}".format(i, 8, i * 8))
print("{0} * {1} = {2}".format(i, 9, i * 9))
print("{0} * {1} = {2}".format(i, 10, i * 10))
i=2
print("{0} * {1} = {2}".format(i, 1, i * 1))
print("{0} * {1} = {2}".format(i, 2, i * 2))
print("{0} * {1} = {2}".format(i, 3, i * 3))
print("{0} * {1} = {2}".format(i, 4, i * 4))
print("{0} * {1} = {2}".format(i, 5, i * 5))
print("{0} * {1} = {2}".format(i, 6, i * 6))
print("{0} * {1} = {2}".format(i, 7, i * 7))
print("{0} * {1} = {2}".format(i, 8, i * 8))
print("{0} * {1} = {2}".format(i, 9, i * 9))
print("{0} * {1} = {2}".format(i, 10, i * 10))
```

```
i=3
...
i=4
...
i=5
...
i=6
...
i=7
...
i=8
...
i=9
print("{0} * {1} = {2}".format(i, 1, i * 1))
print("{0} * {1} = {2}".format(i, 2, i * 2))
print("{0} * {1} = {2}".format(i, 3, i * 3))
print("{0} * {1} = {2}".format(i, 4, i * 4))
print("{0} * {1} = {2}".format(i, 5, i * 5))
print("{0} * {1} = {2}".format(i, 6, i * 6))
print("{0} * {1} = {2}".format(i, 7, i * 7))
print("{0} * {1} = {2}".format(i, 8, i * 8))
print("{0} * {1} = {2}".format(i, 9, i * 9))
print("{0} * {1} = {2}".format(i, 10, i * 10))
```

BT2. Phương án 2: Sử dụng cấu trúc *for*

Chương trình BT3.2.2

```
for i in range(1,11):  
    for j in range(1, 11):  
        print("{0} * {1} = {2}".format(i, j, i * j))
```

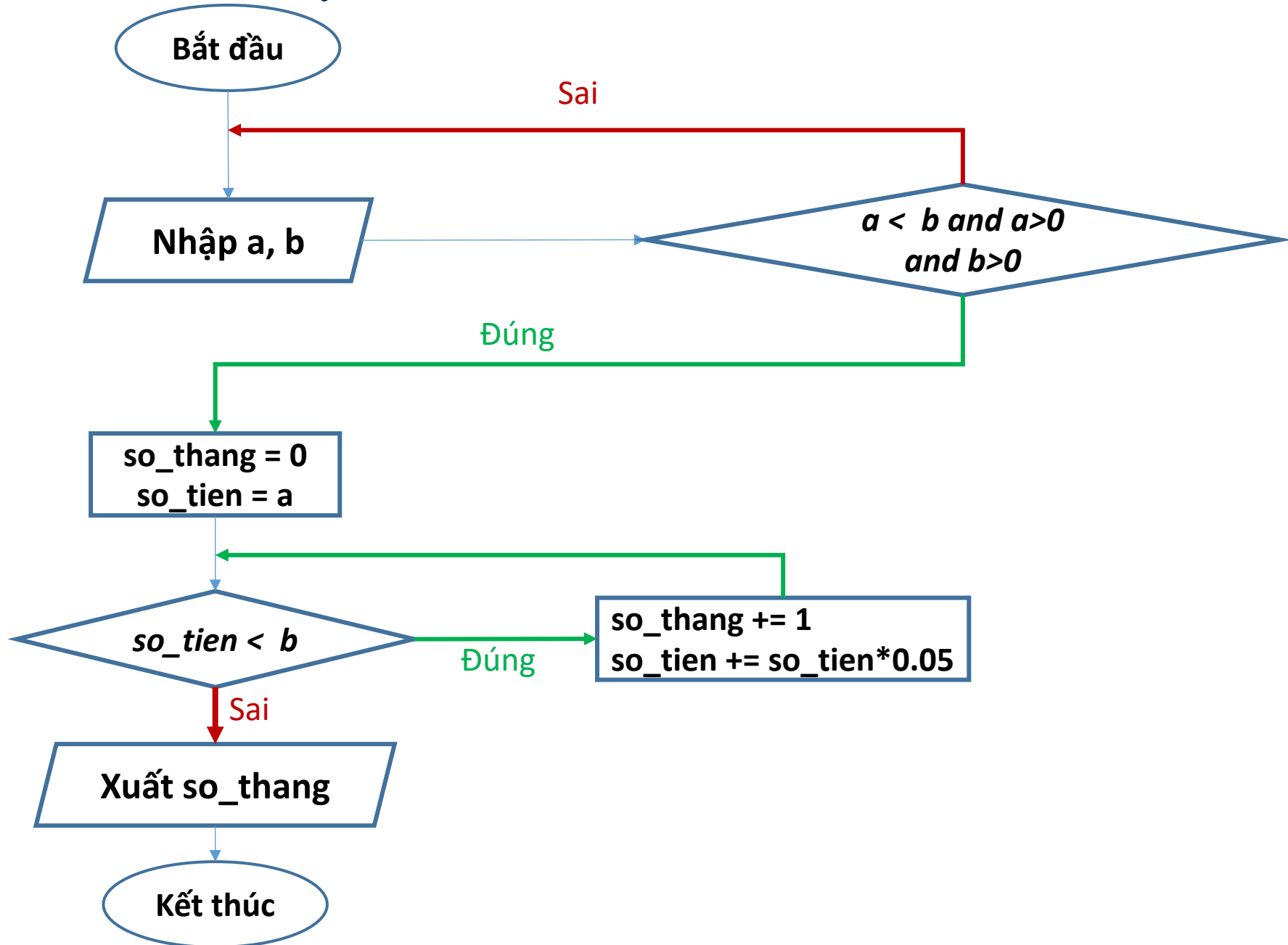
Bài toán 3.3

- ❑ Xây dựng chương trình để giải bài toán tiết kiệm: Giả sử chúng ta có số tiền là a gửi vào một ngân hàng. Hỏi sau bao nhiêu tháng ta thu được số tiền là b ($b > a$) biết rằng lãi suất hàng tháng là 5%.
- ❑ Xác định yêu cầu và cấu trúc dữ liệu
 - **Dữ liệu vào:** Biến thực a và b (số tiền gửi và thu được).
 - **Dữ liệu ra:** Số tháng cần gửi.

Bài toán 3 (tt)

- Cách giải:
 - Số tiền có được sau 1 tháng là số tiền ban đầu a + 5% số tiền a . \Rightarrow Số tiền sau 1 tháng là **$a*1.05$**
- Công việc lặp:
 - $a = a + a * 0.05 = a*1.05$
 - $t = t + 1$
- Điều kiện dừng: Tiền thu được lớn hơn b .

Lưu đồ thuật toán



Bài toán 3 (tt)

Chương trình BT3.3

```
while True:
    a = float(input('Số tiền gửi: '))
    if a > 0: break
while True:
    b = float(input('Số tiền muốn có: '))
    if b > a: break
so_tien = a
so_thang = 0
while so_tien < b:
    so_tien += so_tien * 0.05
    so_thang += 1
print("Cần {0} tháng để gửi tiết kiệm số tiền {1} đạt được lớn hơn hoặc bằng {2}".format(so_thang,a, b))
```

Kết quả

Số tiền gửi: 1000000

Số tiền muốn có: 20000000

Cần 62 tháng để gửi tiết kiệm số tiền 1000000.0 đạt được lớn hơn hoặc bằng 20000000.0

3.2 Cấu trúc lặp *while*

❑ Những chương trình *lặp đi lặp lại nhiều lần* với một điều kiện nào đó, khi đó ta nên sử dụng cấu trúc lặp để thực hiện công việc này. Cấu trúc *while* sẽ thực hiện các công việc (khối lệnh) khi nào còn thỏa mãn điều kiện.

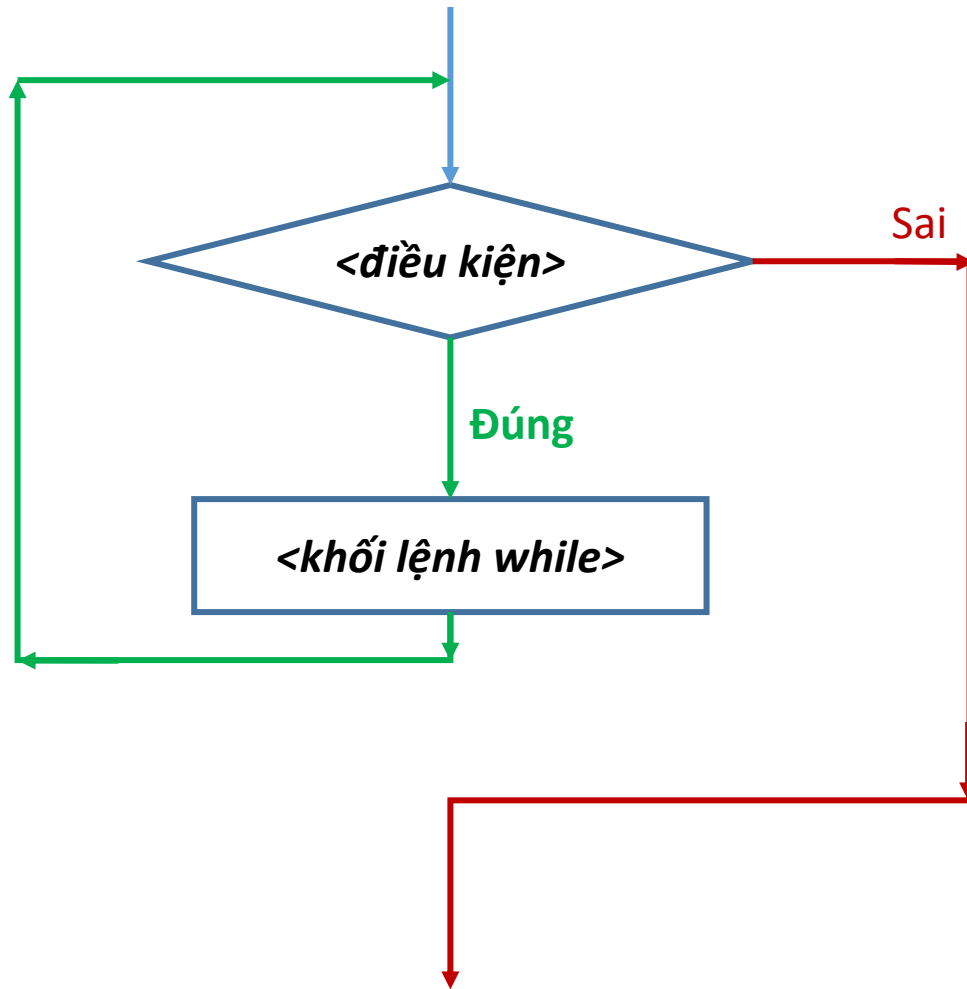
❑ **Cú pháp:**

while <điều kiện>:
 <khối lệnh *while*>

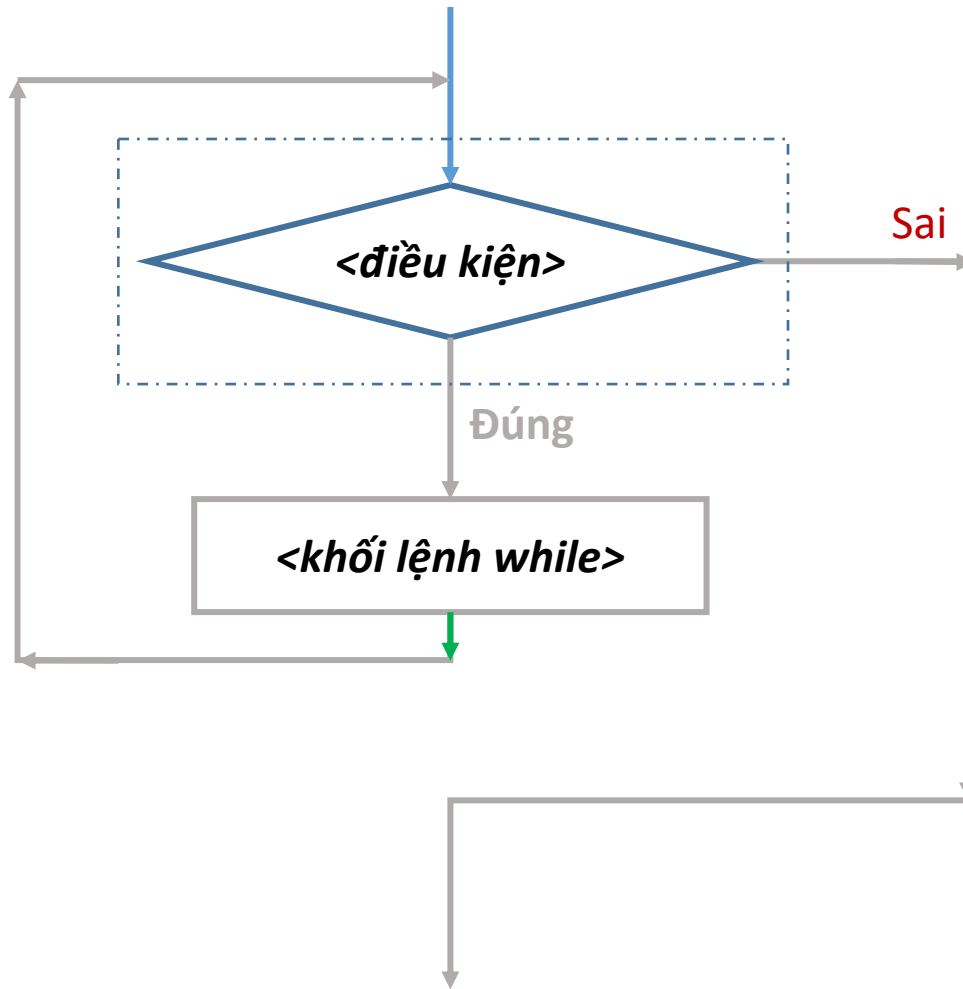
❑ Trong đó:

- *<điều kiện>*: Là một biểu thức logic chỉ điều kiện cho kết quả là Đúng (True) hoặc Sai (False)
- *<Khối lệnh while>*: gồm một hoặc nhiều câu lệnh được chạy khi <điều kiện> là True. Các câu lệnh trong khối lệnh có lẽ trái dịch 1 khoảng so với lề trái.

Lưu đồ thuật toán cấu trúc *while*



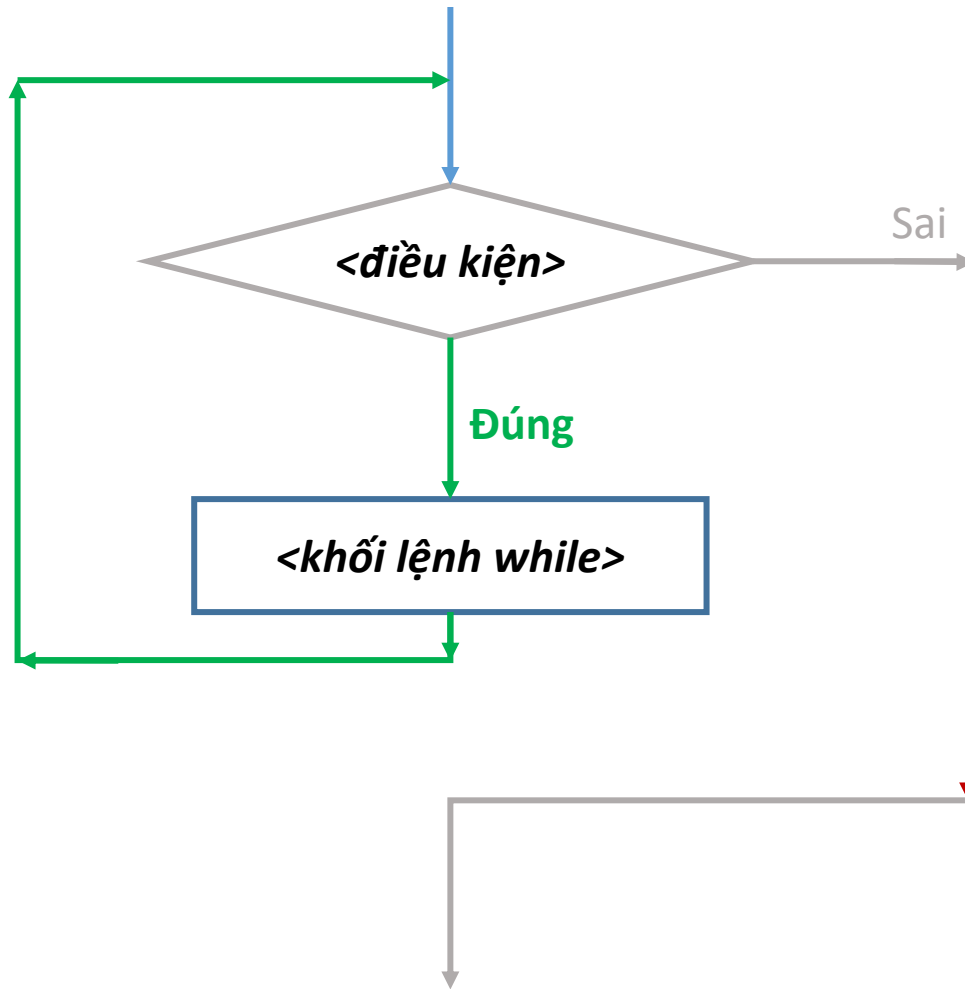
Lưu đồ thuật toán cấu trúc *while*



Bước 1:

Kiểm tra *<điều kiện>* Nếu True thì qua Bước 2. Nếu False thì qua Bước 3.

Lưu đồ thuật toán cấu trúc *while*



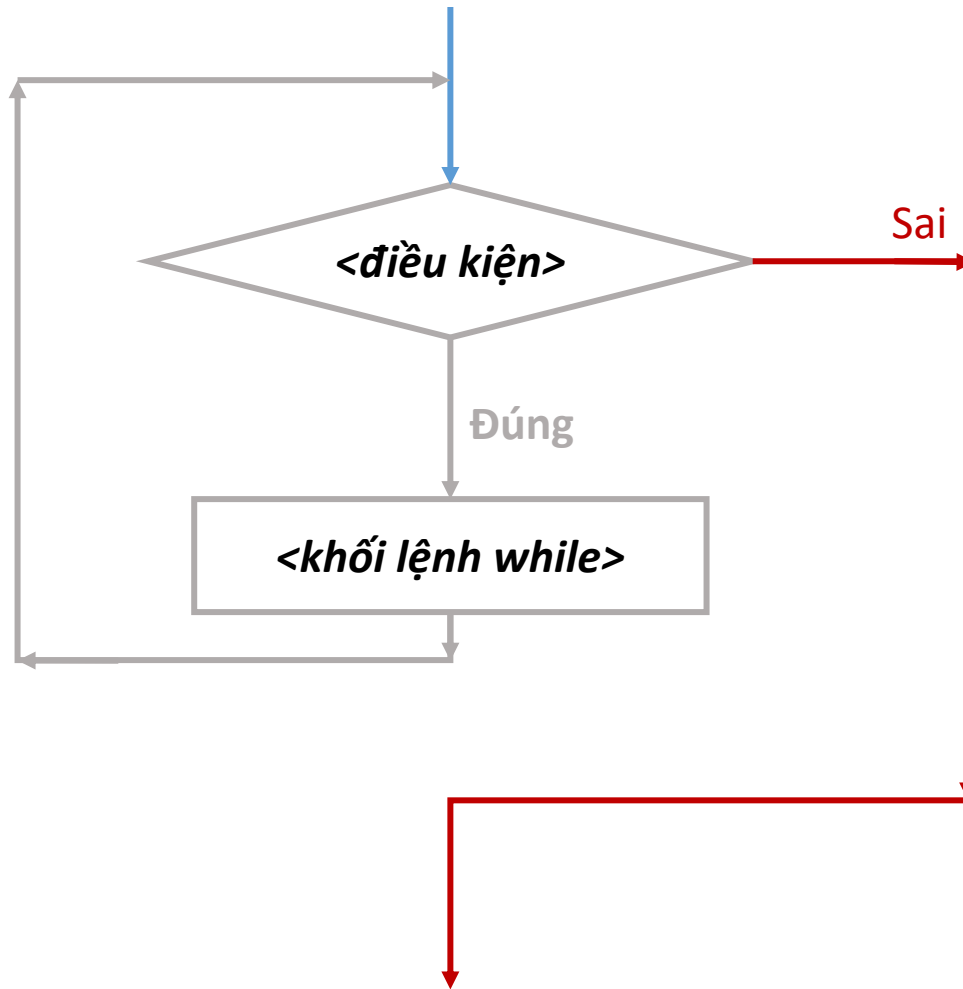
Bước 1:

Kiểm tra <điều kiện> Nếu True thì qua Bước 2. Nếu False thì qua Bước 3.

Bước 2:

Thực hiện các lệnh trong <khối lệnh while>, sau đó quay lại Bước 1.

Lưu đồ thuật toán cấu trúc *while*



Bước 1:

Kiểm tra *<điều kiện>* Nếu True thì qua Bước 2. Nếu False thì qua Bước 3.

Bước 3:

Kết thúc vòng lặp *while*

Bài toán 3.4

- Xây dựng chương trình hiển thị ra màn hình các tháng trong năm

```
i = 1
while i < 13:
    print("Tháng", i)
    i += 1
```

Bài toán 3.5

- Viết chương trình đếm các số ước số nguyên dương n

3.3 Cấu trúc lặp *for*

❑ **Cấu trúc lặp** *for* giúp thao tác với nhiều kiểu dữ liệu khác nhau một cách thuận tiện và dễ dàng.

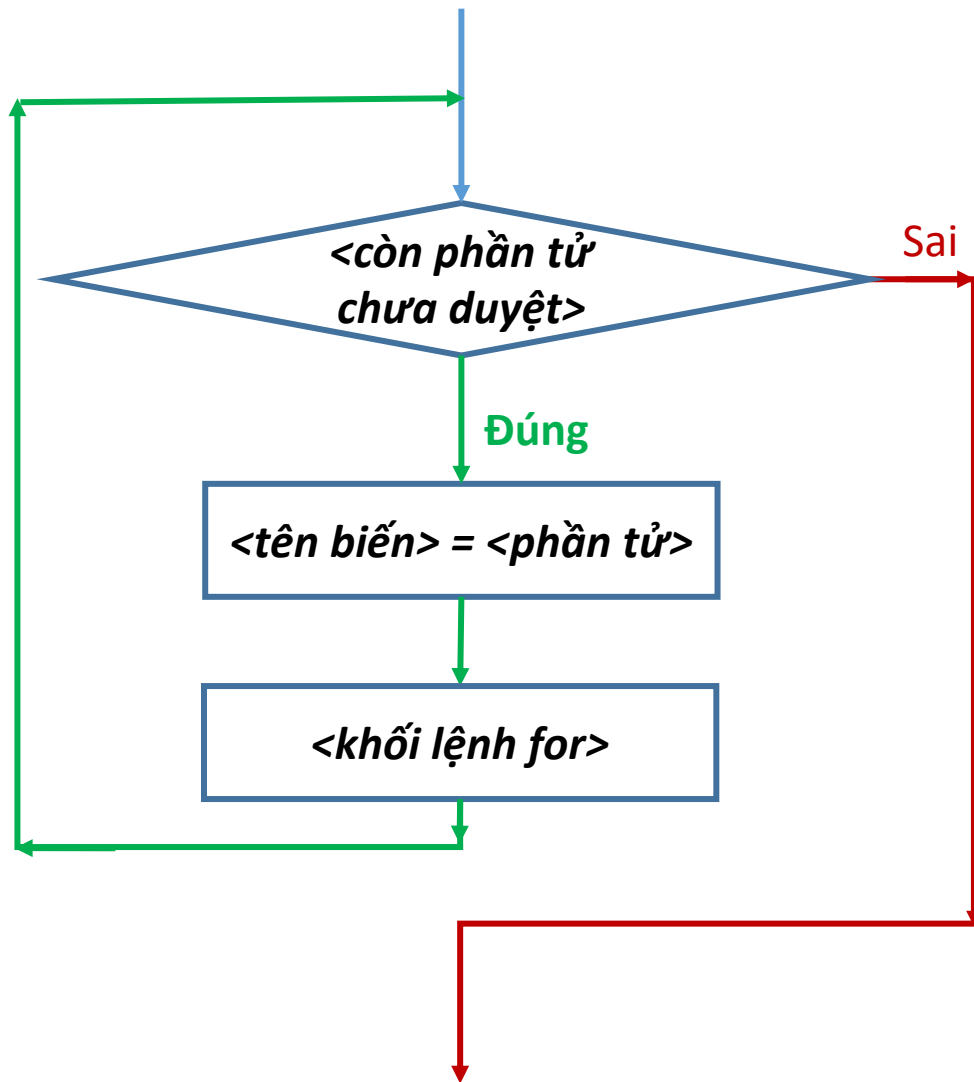
❑ **Cú pháp**

for *<tên biến>* **in** *<tập hợp>*:
<khối lệnh for>

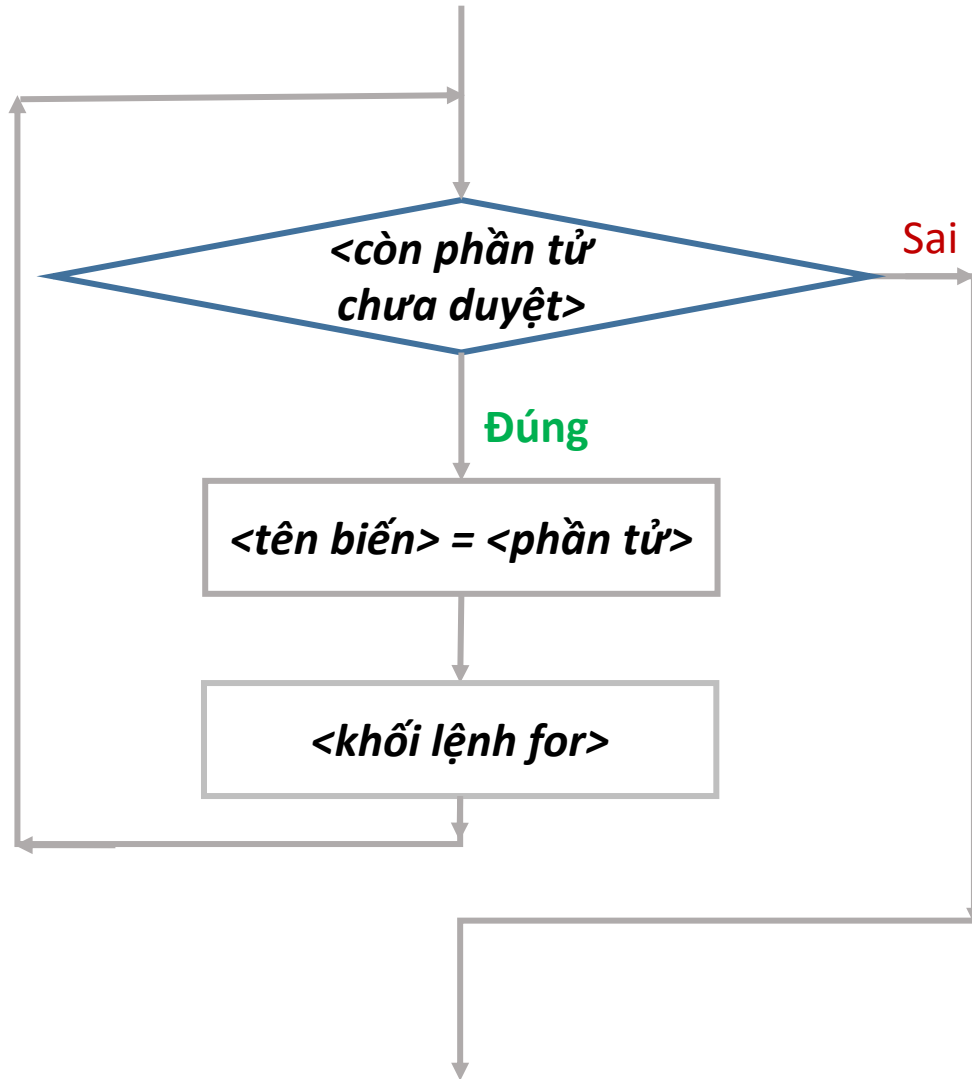
❑ Trong đó:

- *<tên biến>*: là một tên do người dùng đặt
- *<tập hợp>*: Là một trong các phần tử của 1 list, tuple, chuỗi,...
- *<khối lệnh for>*: bao gồm 1 hoặc nhiều câu lệnh được chạy khi duyệt qua các phần tử, khối lệnh này cách 1 khoảng so với câu lệnh *for*

Lưu đồ thuật toán cấu trúc *for*



Lưu đồ thuật toán cấu trúc *for*

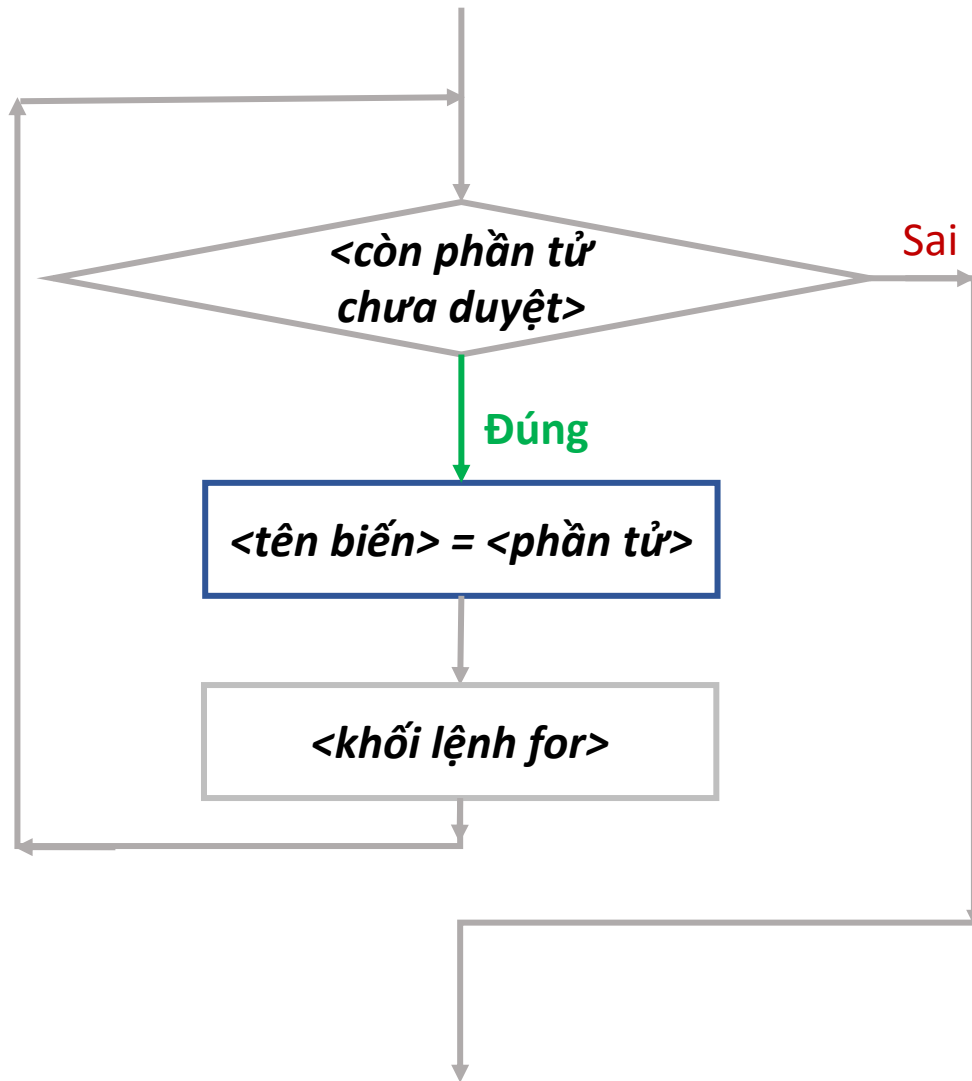


Bước 1:

Kiểm tra trong tập hợp có phần tử chưa được duyệt hay không?

Nếu còn (True) thì thực hiện **Bước 2**; Ngược lại (False), chuyển sang **Bước 5**.

Lưu đồ thuật toán cấu trúc *for*



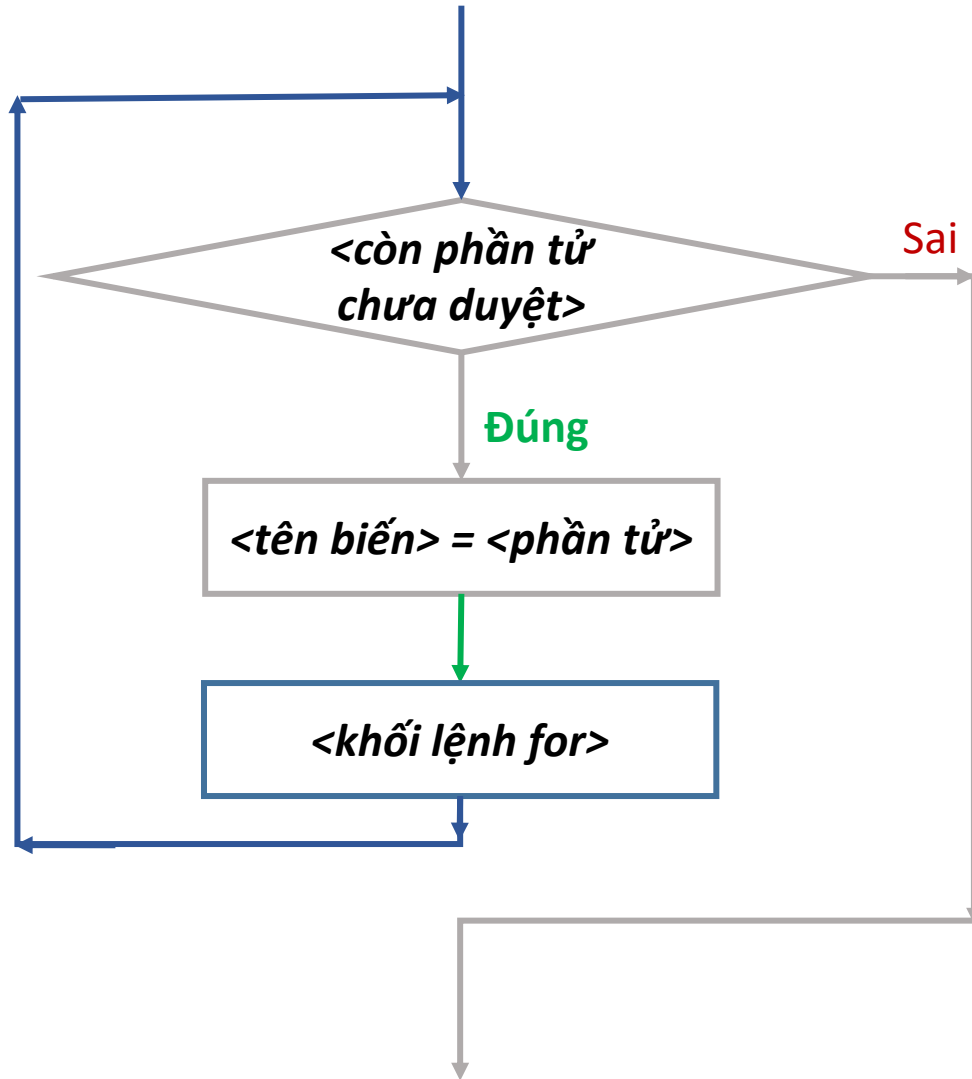
Bước 1:

Kiểm tra trong tập hợp có phần tử chưa được duyệt hay không?

Nếu còn (True) thì thực hiện **Bước 2**; Ngược lại (False), chuyển sang **Bước 5**.

Bước 2: Lấy ra phần tử chưa được duyệt trong <tập hợp> và gán cho <tên biến>.

Lưu đồ thuật toán cấu trúc *for*



Bước 1:

Kiểm tra trong tập hợp có phần tử chưa được duyệt hay không?

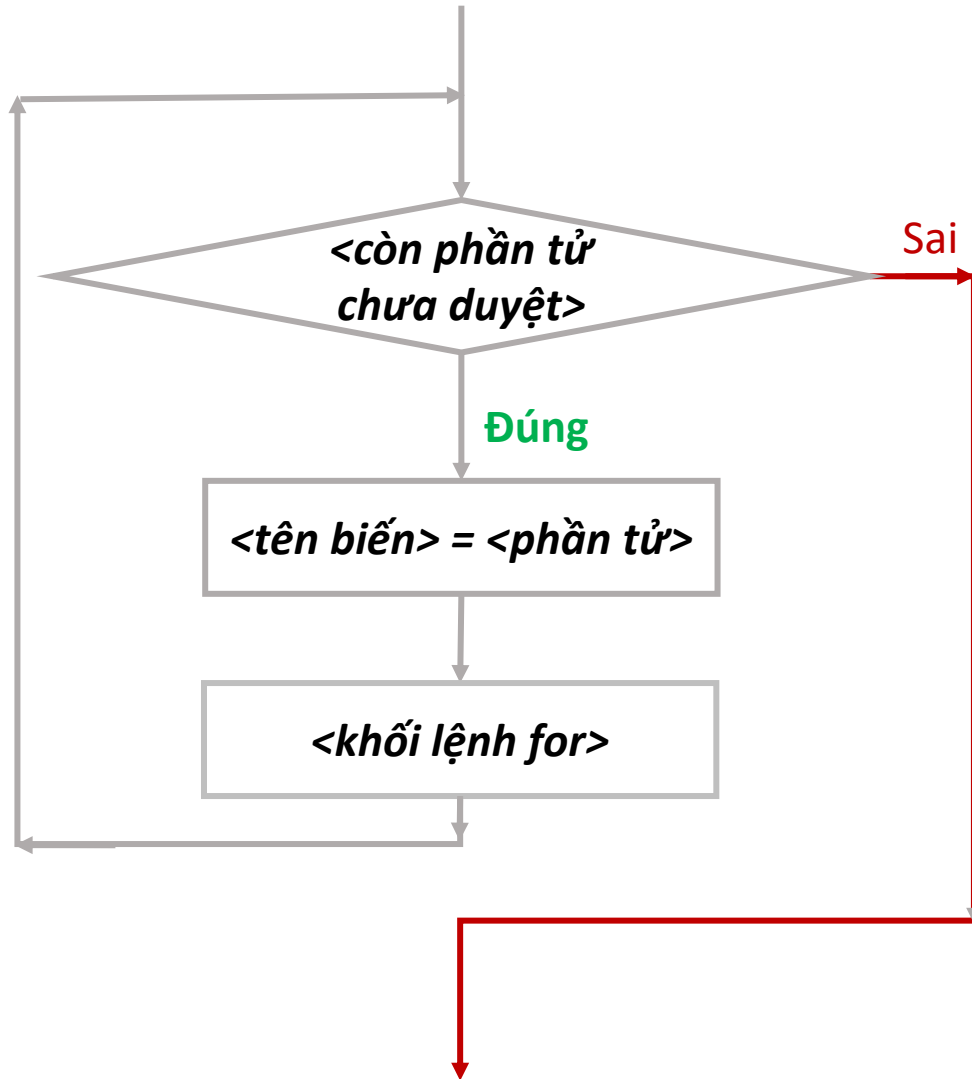
Nếu còn (True) thì thực hiện **Bước 2**; Ngược lại (False), chuyển sang **Bước 5**.

Bước 2: Lấy ra phần tử chưa được duyệt trong <tập hợp> và gán cho <tên biến>.

Bước 3: Thực hiện <khối lệnh for>.

Bước 4: Quay lên thực hiện lại Bước 1.

Lưu đồ thuật toán cấu trúc *for*



Bước 1:

Kiểm tra trong tập hợp có phần tử chưa được duyệt hay không?

Nếu còn (True) thì thực hiện **Bước 2**; Ngược lại (False), chuyển sang **Bước 5**.

Bước 5: Kết thúc câu lặp *for*

Ví dụ 3.3.1

- Viết chương trình tính tổng các phần tử từ 1 đến 5, sau đó in kết quả ra màn hình.

```
>>> for i in [1, 2, 3, 4, 5]:  
...     s = s + i  
...  
>>> print("Tổng = ",s)  
Tổng = 15
```

```
>>> s = 0  
>>> for i in range(1, 6):  
...     s = s + i  
...  
>>> print("Tổng =", s)  
Tổng = 15
```

Ví dụ 3.3.2

- Viết chương trình hiển thị ra màn hình 100 số tự nhiên đầu tiên.

```
>>> for i in range(100):  
...     print(i)  
...  
0  
1  
2  
...  
96  
97  
98  
99  
>>>
```

Hàm `range()`, sử dụng để tạo ra một danh sách, dãy số.

Hàm range()

❑ Hàm range(), sử dụng để tạo ra một danh sách, dãy số.

❑ **Cú pháp:**

range([<start>,] <end> [,<step>])

❑ Trong đó:

- Hàm range sẽ tạo ra một dãy các phần tử có giá trị từ <start> đến <end> theo một bước nhảy giữa các phần tử là <step>
- <start>: chỉ định phần tử bắt đầu, mặc định là 0.
- <end>: chỉ định phần tử kết thúc, bắt buộc phải có phần tử này.
- <step>: chỉ định bước nhảy giữa hai số <start> và <end>, mặc định là 1.

Ví dụ 3.3.3

- Sử dụng hàm range tạo ra một dãy số.

```
>>> print(range(0, 10))  
range(0, 10)
```

```
>>> print(list(range(9)))  
[0, 1, 2, 3, 4, 5, 6, 7, 8]
```

```
>>> print(list(range(2, 5)))  
[2, 3, 4]
```

```
>>> print(list(range(0, 15, 5)))  
[0, 5, 10]
```


3.4 Lệnh *break* và *continue*

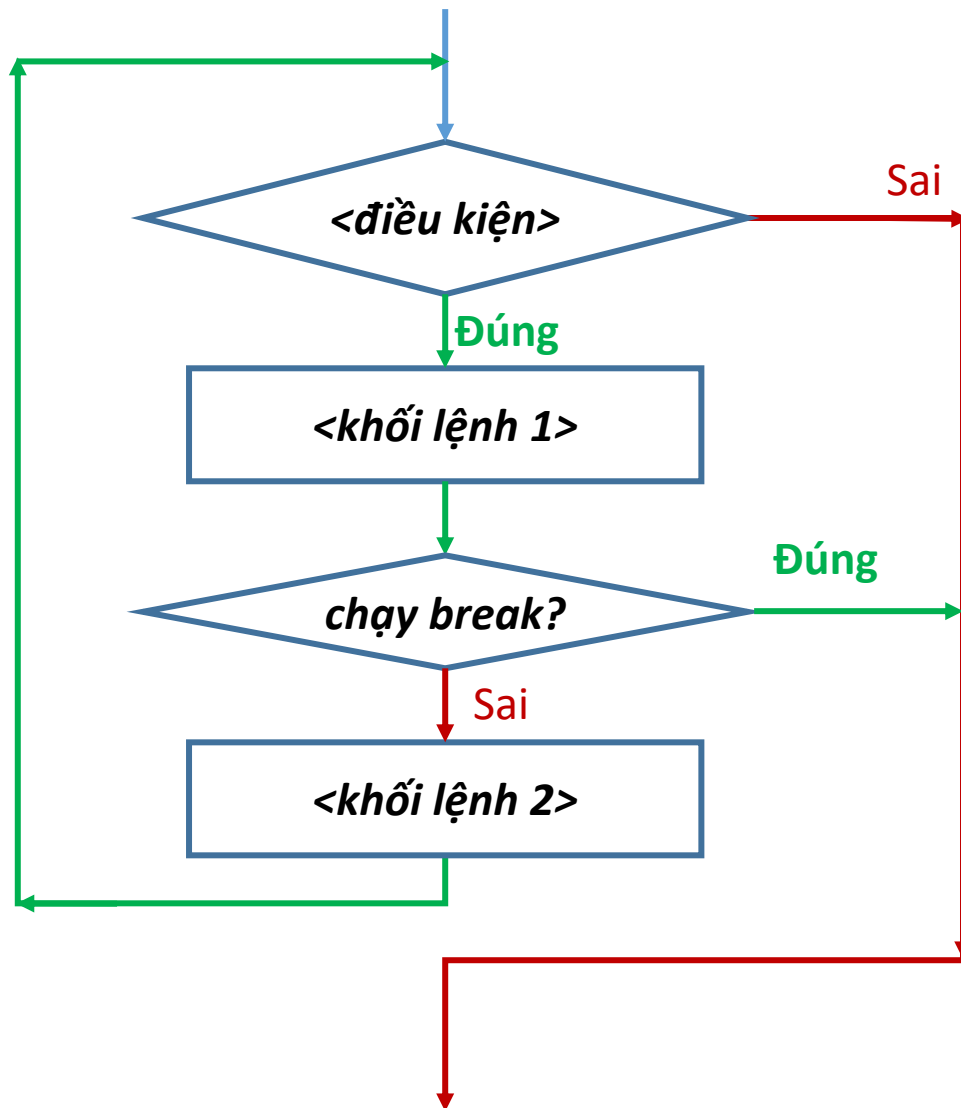
- Được sử dụng bên trong vòng lặp nhằm chấm dứt việc lặp lại
- Thoát khỏi cấu trúc lặp while, for.
- Bỏ qua phần còn lại của khối lệnh.

3.4.1 Lệnh break

- Lệnh break để thoát khỏi cấu trúc lặp ngay lập tức mà không cần kiểm tra điều kiện.
- Nếu lệnh break nằm trong một cấu trúc lặp lồng nhau thì nó sẽ thoát khỏi cấu trúc lặp chứa nó.
- Cú pháp:

break

Lưu đồ thuật toán cấu trúc lặp chứa lệnh break



Ví dụ 3.3.4

- Viết chương trình đọc tất cả các ký tự trong chuỗi “Python là ngôn ngữ lập trình” và kiểm tra điều kiện, nếu chương trình đọc được chữ cái n đầu tiên trong chuỗi thì dừng chương trình, ngược lại thì in ra ký tự ra màn hình.

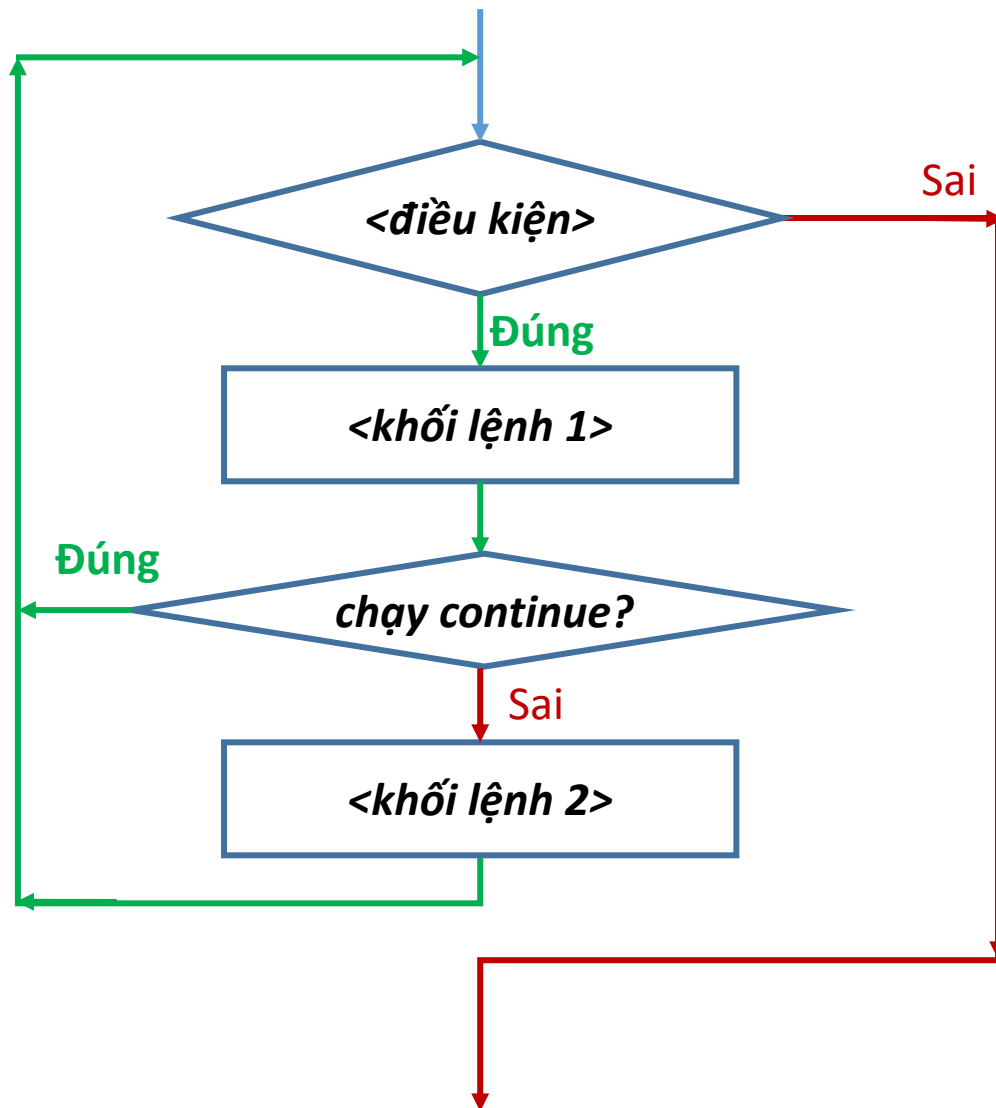
Ví dụ 3.3.4 (tt)

```
>>> for i in "Python là ngôn ngữ lập trình":  
...     print(i)  
...     if i == "n":  
...         break  
...  
P  
y  
t  
h  
o  
n
```

3.4.3 Lệnh continue

- Lệnh continue bỏ qua các lệnh phía sau nó trong khối lệnh lặp và quay lên đầu kiểm tra điều kiện lặp ngay lập tức để tiếp tục lần lặp kế tiếp
- Cú pháp: **continue**

Lưu đồ thuật toán cấu trúc lặp chứa lệnh continue



Ví dụ 3.3.5

- Viết chương trình hiển thị ra màn hình các số từ 9 đến 0 và kiểm tra điều kiện, nếu bằng 5 thì chương trình không hiển thị mà quay lại cấu trúc lặp.

Ví dụ 3.3.5 (tt)

```
>>> a = 10
>>> while a > 0:
...     a -= 1
...     if a == 5:
...         continue
...     print("Giá trị biến hiện tại là:", a)
...
Giá trị biến hiện tại là: 9
Giá trị biến hiện tại là: 8
Giá trị biến hiện tại là: 7
Giá trị biến hiện tại là: 6
Giá trị biến hiện tại là: 4
Giá trị biến hiện tại là: 3
Giá trị biến hiện tại là: 2
Giá trị biến hiện tại là: 1
Giá trị biến hiện tại là: 0
```

Bài tập

□ **Bài tập 1:** Viết chương trình cho người dùng nhập 1 số nguyên n , hãy kiểm tra xem số vừa nhập có phải là số nguyên tố hay không.

- Số nguyên tố là số tự nhiên lớn hơn 1 không phải là tích của hai số tự nhiên nhỏ hơn. Nói cách khác, số nguyên tố là những số chỉ có đúng hai ước số là 1 và chính nó

□ **Bài tập 2:** Viết chương trình cho phép nhập hai số nguyên dương x và n từ bàn phím. Tính $S(x,n)$ và hiển thị ra màn hình $S(n)$. Với:

$$S(x, n) = x^2 + x^4 + \dots + x^{2n}$$

Bài tập (tt)

□ **Bài tập 3:** Một sinh viên tiết kiệm được một số tiền 8,000,000 đồng. Sinh viên lập 1 sổ tiết kiệm và gửi vào ngân hàng với lãi suất 9% 1 tháng.

- Hãy viết chương trình tính xem sau 5 năm sinh viên này sẽ nhận được số tiền là bao nhiêu (được làm tròn đến hàng nghìn) với giả định rằng trong thời gian này sinh viên không thực hiện tất toán sổ tiết kiệm trên.