



LẬP TRÌNH PYTHON

CẤU TRÚC VÒNG LẶP

Giảng viên: Nguyễn Thái Khánh

GIỚI THIỆU CẤU TRÚC VÒNG LẶP

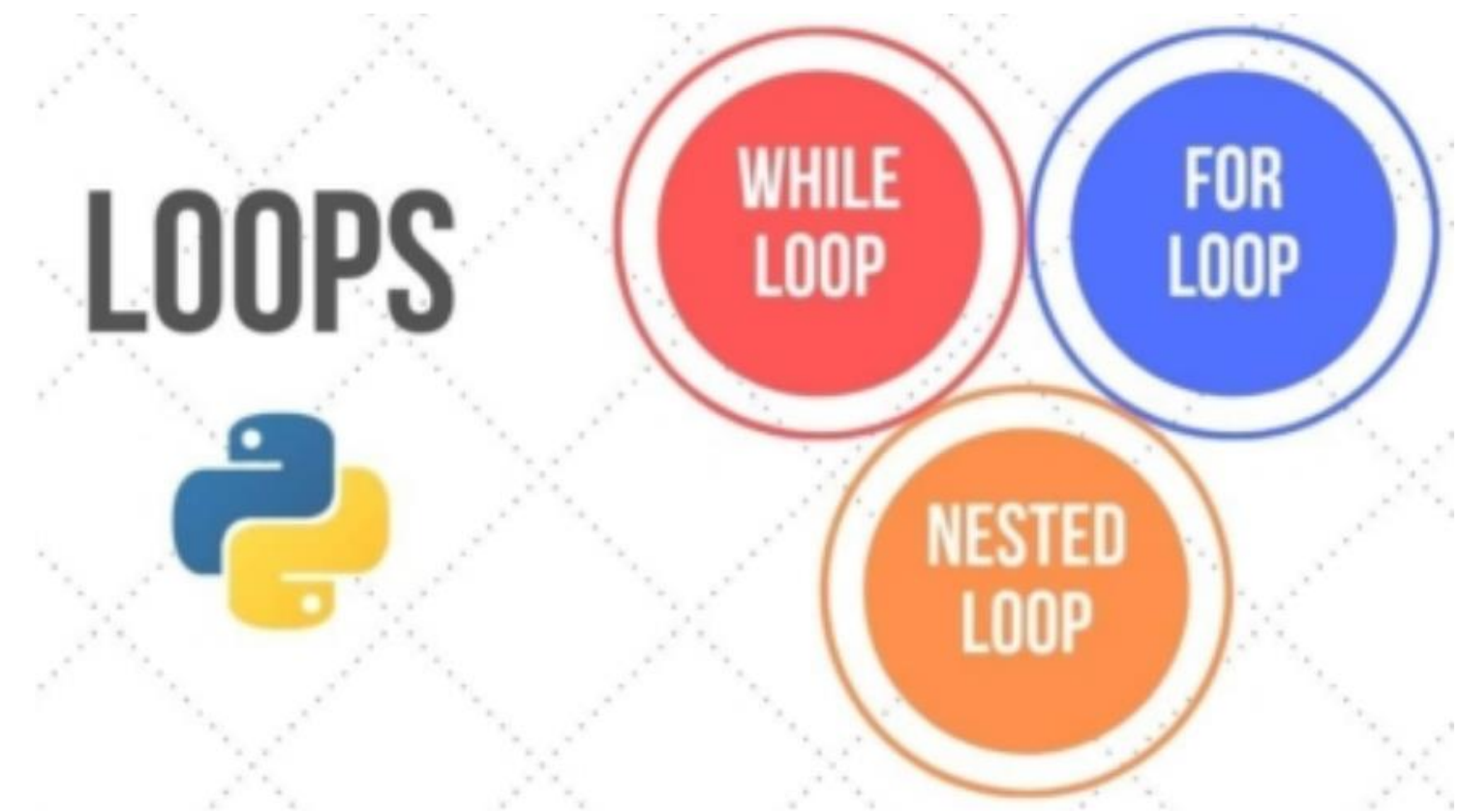
VÒNG LẶP WHILE: Vòng lặp được điều khiển bởi điều kiện

VÒNG LẶP FOR: Vòng lặp được điều khiển bằng số lần lặp

VÒNG LẶP LỒNG NHAU

****) Yêu cầu: Viết một chương trình thực hiện cùng 1 nhiệm vụ nhiều lần***

- Nếu sao chép code:
 - + Làm cho chương trình lớn
 - + Tốn thời gian
 - + Có thể cần phải sửa lỗi ở nhiều nơi



⇒ Cấu trúc vòng lặp giải quyết vấn đề trên, cho máy tính lặp lại đoạn mã khi cần thiết:

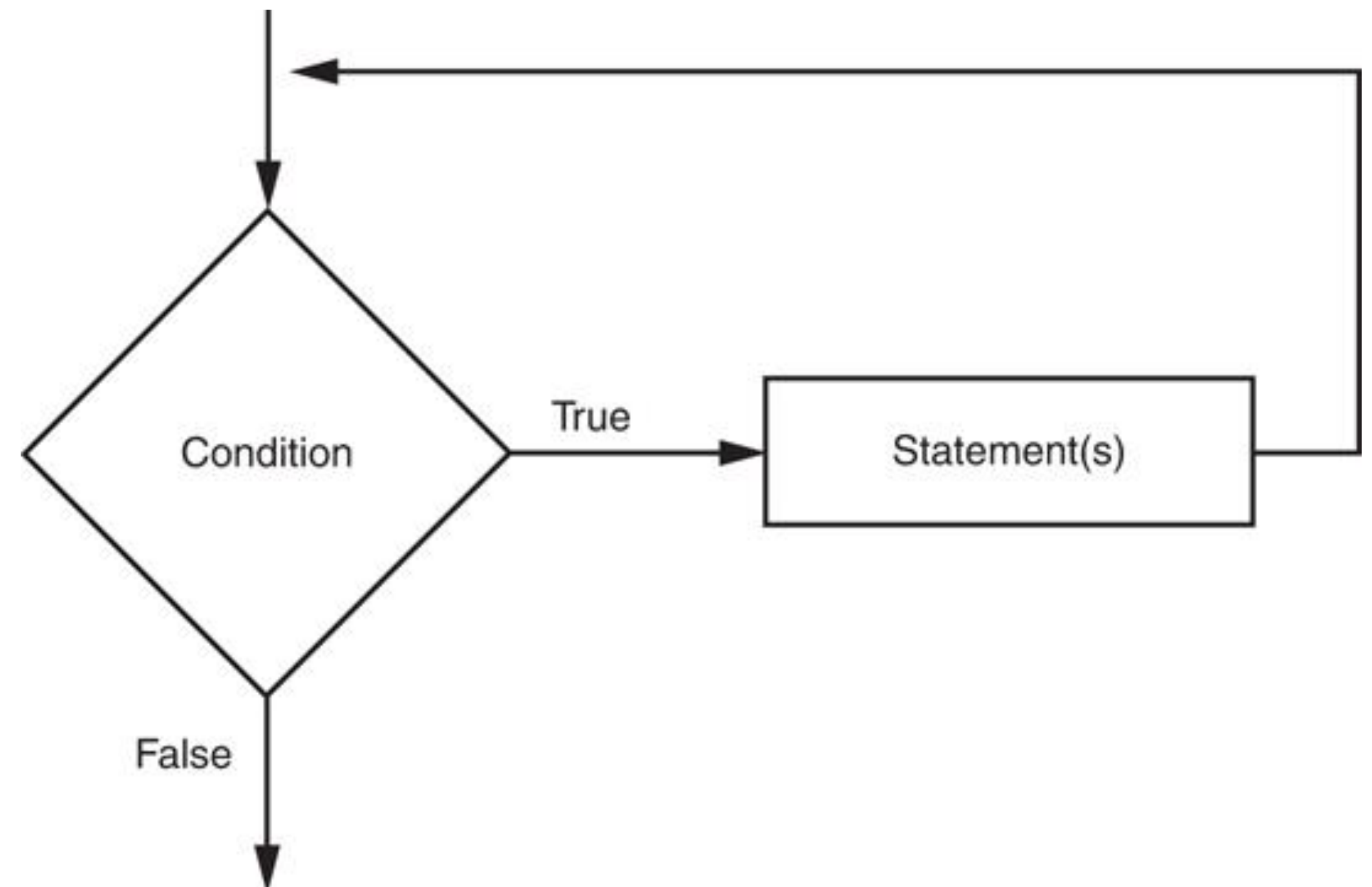
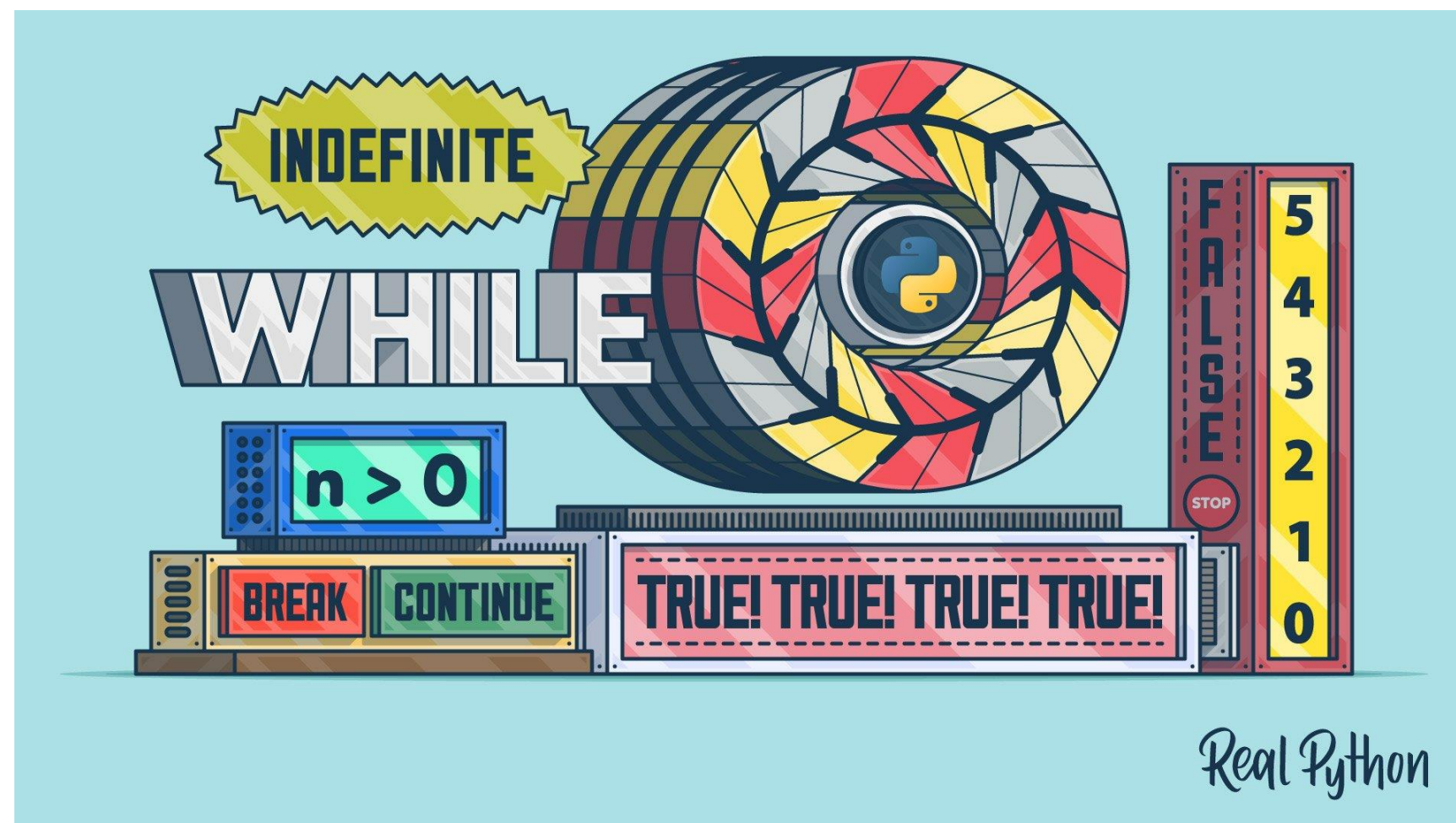
Bao gồm các vòng lặp được điều khiển bằng điều kiện, và vòng lặp được điều khiển bằng số lần lặp

VÒNG LẶP WHILE

Vòng lặp được điều khiển bởi điều kiện

- **Vòng lặp While: khi điều kiện đúng, thực hiện điều gì đó**
- Hai phần:
 - + Điều kiện được kiểm tra để xác định giá trị đúng hay sai
 - + Các lệnh được lặp lại miễn là điều kiện đúng

`while conditions:
statements`



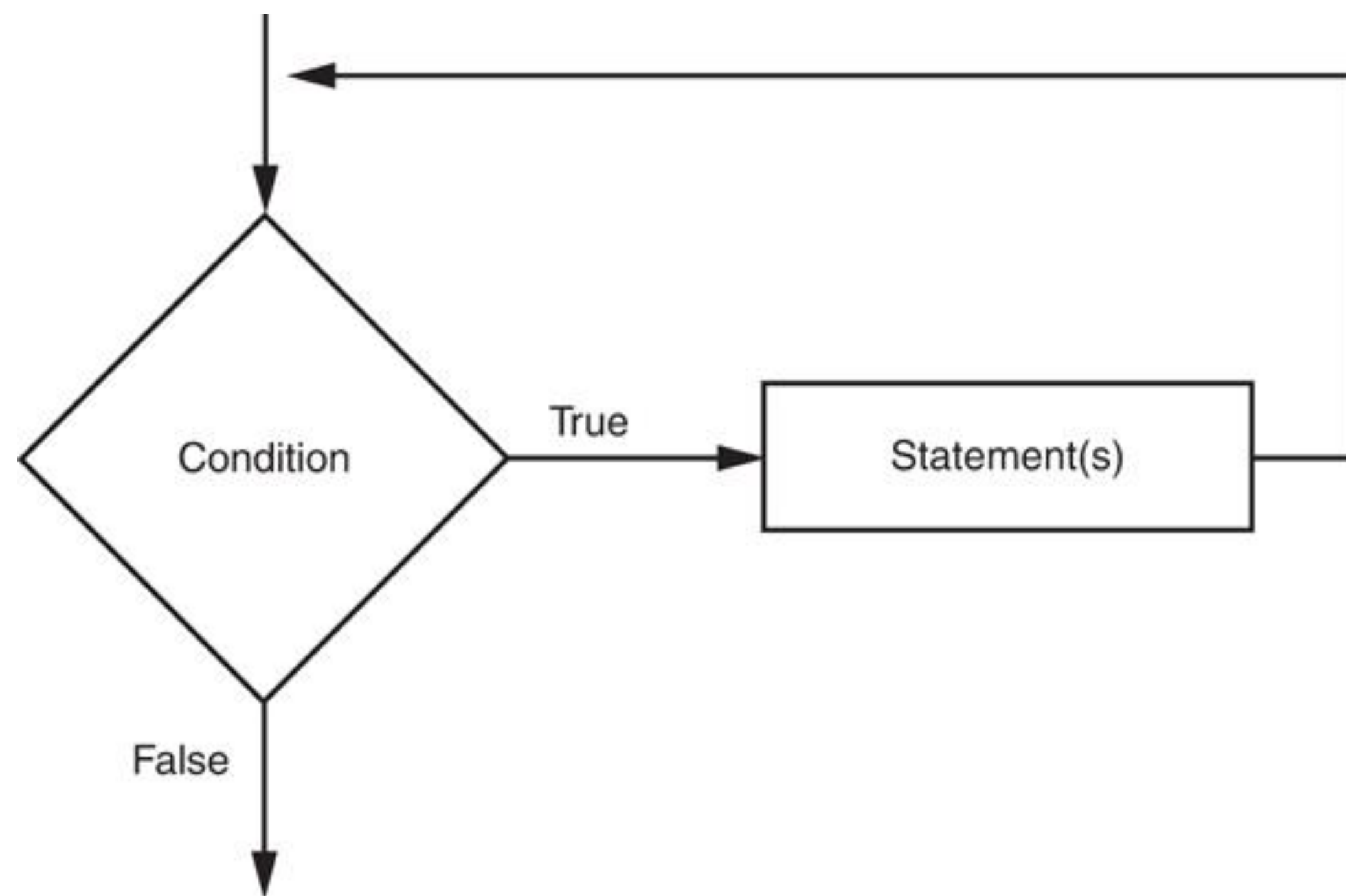
Vậy làm sao để vòng lặp *while* có thể ngừng thực hiện??



VÒNG LẶP WHILE

Vòng lặp được điều khiển bởi điều kiện

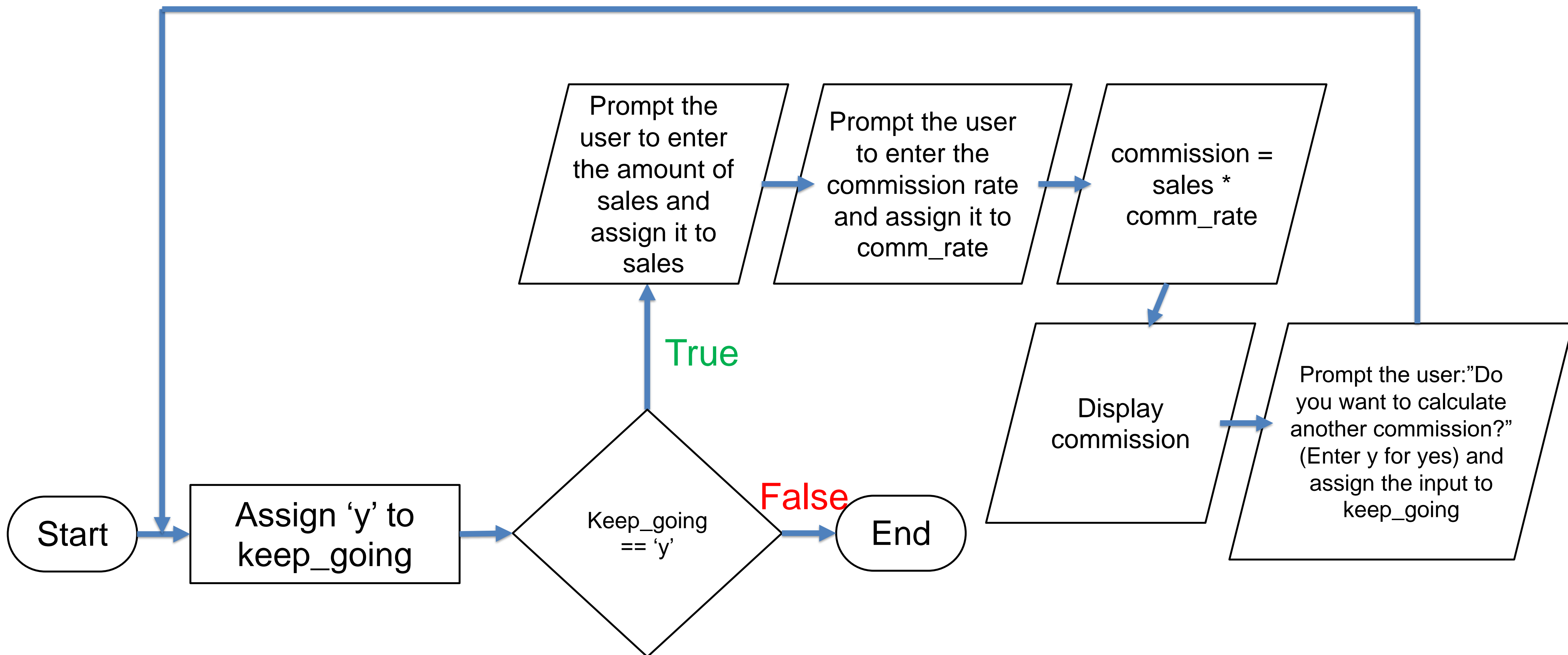
- Để vòng lặp dừng thực hiện, cần phải có ***điều gì đó xảy ra bên trong*** vòng lặp để làm cho ***điều kiện trở nên sai***
- Mỗi một vòng lặp: một lần thực hiện phần thân của vòng lặp
- Vòng lặp while được gọi là vòng lặp kiểm tra trước (pretest loop):
 - + Kiểm tra điều kiện trước khi thực hiện một lần lặp
 - + Sẽ không bao giờ thực hiện nếu điều kiện sai



VÒNG LẶP WHILE

Vòng lặp được điều khiển bởi điều kiện

**) Vừa học Tiếng Anh, vừa phân tích yêu cầu bài toán ^^*



Vậy liệu có một vòng lặp, lặp đến vô tận
hay không??



INFINITE LOOPS

- Thông thường với Vòng lặp while: phải chứa bên trong nó một cách để dừng lại
- Vòng lặp vô hạn: vòng lặp không có cách nào để dừng lại:
 - + Nó sẽ lặp lại khối chương trình bên trong cho đến khi bị ngắt (Ctrl+C)
 - + Thường xảy ra khi quên thêm câu lệnh để dừng vòng lặp



VÒNG LẶP FOR

Vòng lặp được điều khiển bởi số lần lặp


- Vòng lặp được điều khiển bằng số lần lặp:
 - + Sử dụng câu lệnh *for* để viết vòng lặp được biểu diễn bằng số lần lặp
 - + Được thiết kế để làm việc với chuỗi các mục dữ liệu
 - + Lặp một lần cho mỗi mục trong chuỗi

for variable in [val1, val2, etc.]:
statements

- + Biến mục tiêu: là đối tượng của việc gán giá trị ở đầu mỗi vòng lặp




Vòng lặp
thứ nhất




```
for num in [1, 2, 3, 4, 5]:  
    print(num)
```

Vòng lặp
thứ hai




```
for num in [1, 2, 3, 4, 5]:  
    print(num)
```

Vòng lặp
thứ ba




```
for num in [1, 2, 3, 4, 5]:  
    print(num)
```

Vòng lặp
thứ tư



```
for num in [1, 2, 3, 4, 5]:  
    print(num)
```

Vòng lặp
thứ năm



```
for num in [1, 2, 3, 4, 5]:  
    print(num)
```


VÒNG LẶP FOR

Vòng lặp được điều khiển bởi số lần lặp

*) Sử dụng hàm *range* với vòng lặp *for*:

- Hàm range đơn giản hóa quá trình viết vòng lặp for
- range trả về một đối tượng có thể lặp
- Đối tượng có thể lặp (Iterable): chứa một chuỗi các giá trị có thể được lặp qua
- Đặc điểm của *range*:
 - + Một tham số: được sử dụng làm giới hạn kết thúc
 - + Hai tham số: giá trị bắt đầu và giới hạn kết thúc
 - + Ba tham số: tham số thứ ba là giá trị bước (step value)

the first element
↓
`range(0, 10, 2)`
↑
the last element

the "step"

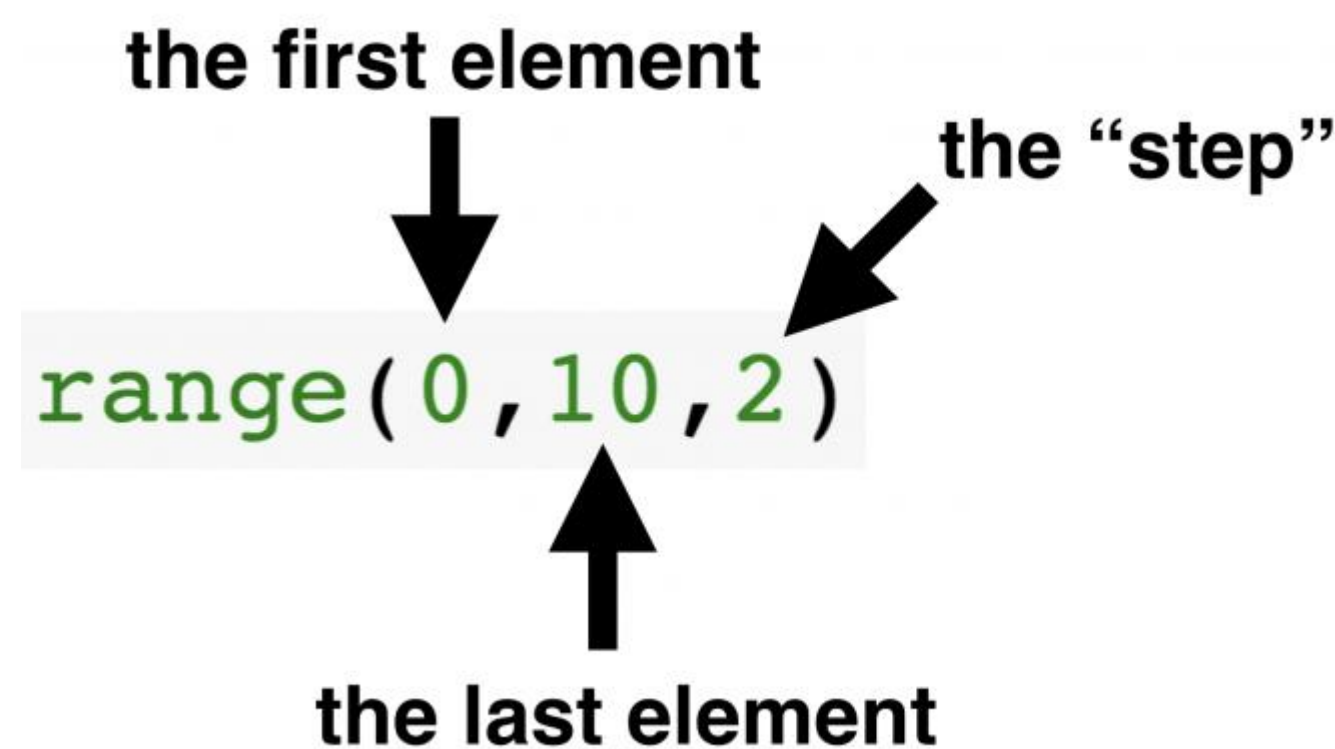
```
for x in range(1, 11):  
    print(x)
```

VÒNG LẶP FOR

Vòng lặp được điều khiển bởi số lần lặp

***) Sử dụng hàm *range* với vòng lặp *for*:**

- Hàm range đơn giản hóa quá trình viết vòng lặp for
- range trả về một đối tượng có thể lặp
- Đối tượng có thể lặp (Iterable): chứa một chuỗi các giá trị có thể được lặp qua
- Đặc điểm của *range*:
 - + Một tham số: được sử dụng làm giới hạn kết thúc
 - + Hai tham số: giá trị bắt đầu và giới hạn kết thúc
 - + Ba tham số: tham số thứ ba là giá trị bước (step value)



VÒNG LẶP FOR

Vòng lặp được điều khiển bởi số lần lặp

***) Sử dụng hàm *range* với vòng lặp *for*:**

- Chạy thử chương trình sau và đưa ra nhận xét:

```
for x in range(1, 11):  
    print(x)
```

- Liệu chương trình sau đây có chạy được không??

```
for i in range(10, 0, -1)  
    print(i)
```


Bài tập

Hãy viết một chương trình tính tổng một chuỗi số cho trước, sử dụng vòng lặp while và for

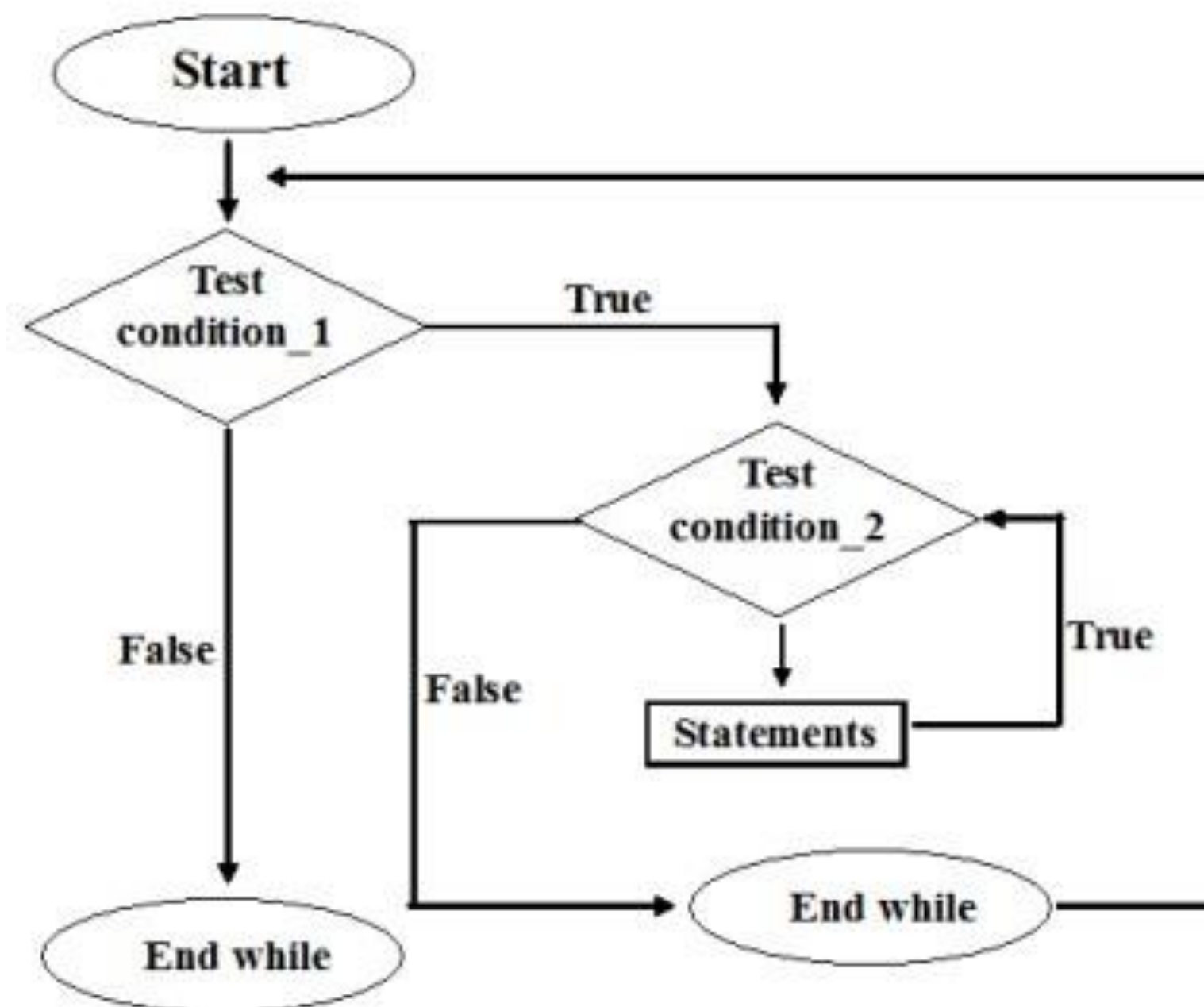
=> *Phân tích bài toán, vẽ đồ thị flowchart, lập trình*

Operator	Example Usage	Equivalent To
<code>+=</code>	<code>x += 5</code>	<code>x = x + 5</code>
<code>-=</code>	<code>y -= 2</code>	<code>y = y - 2</code>
<code>*=</code>	<code>z *= 10</code>	<code>z = z * 10</code>
<code>/=</code>	<code>a /= b</code>	<code>a = a / b</code>
<code>%=</code>	<code>c %= 3</code>	<code>c = c % 3</code>
<code>//=</code>	<code>x //= 3</code>	<code>x = x // 3</code>
<code>**=</code>	<code>y **= 2</code>	<code>y = y ** 2</code>

- Vòng lặp lồng nhau: vòng lặp nằm bên trong một vòng lặp khác

Ví dụ: đồng hồ kim hoạt động như một vòng lặp lồng nhau

- Kim giờ di chuyển một lần cho mỗi mười hai lần di chuyển của kim phút
- Kim giây di chuyển 60 lần cho mỗi lần di chuyển của kim phút



- Những điểm chính về vòng lặp lồng nhau:

- + Vòng lặp bên trong thực hiện tất cả các lần lặp của nó cho mỗi lần lặp của vòng lặp bên ngoài.
- + Vòng lặp bên trong hoàn thành các lần lặp của nó nhanh hơn so với vòng lặp bên ngoài.
- + Tổng số lần lặp trong vòng lặp lồng nhau bằng tích của số lần lặp vòng lặp bên trong và số lần lặp của vòng lặp bên ngoài

*Nếu một chiếc kim giây thực hiện chương trình của mình 997200 lần, vậy kim giờ thực hiện chương trình của nó bao nhiêu lần?? **Sử dụng vòng lặp lồng nhau để kiểm tra (vòng lặp for)***

Chương này đã đề cập đến:

- Cấu trúc vòng lặp, bao gồm:
 - Vòng lặp điều kiện
 - Vòng lặp điều khiển số lần lặp
 - Vòng lặp lồng nhau
 - Vòng lặp vô hạn và cách tránh chúng
- Hàm ``range`` được sử dụng trong vòng lặp ``for``
- Tính toán tổng liên tục và các toán tử gán mở rộng
- Sử dụng dấu hiệu kết thúc để kết thúc vòng lặp



Cảm ơn đã lắng nghe!