การใช้คำสั่ง while loop

วนลูปอย่างมีสไตล์ ควบคุมโค้ดได้ดั่งใจ

รายวิชา การเขียนโปรแกรมคอมพิวเตอร์

สาขาวิชาเทคโนโลยีธุรกิจดิจิทัล | วิทยาลัยการอาชีพขุนหาญ

ผู้สอน: นายอัจฉริยะ มูลจันดา

กลุ่มเป้าหมาย: นักศึกษา ปวส. 1

เขียนโค้ดซ้ำซ้อน (เหนื่อยและเสียเวลา)

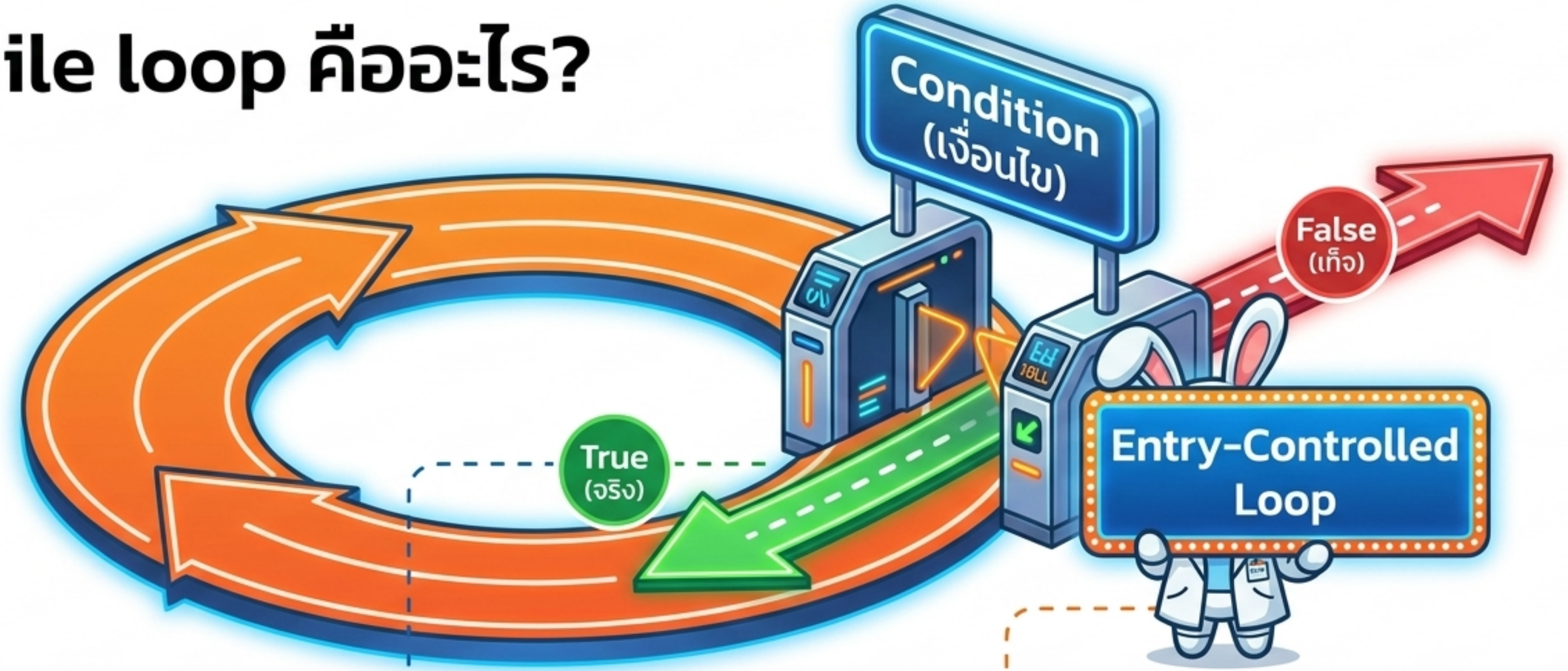
```
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
```

ใช้ Loop บรรทัดเดียวจบ (DRY Concept)

```
int i = 1;
while (i <= 100) {
    printf("Hello\n");
    i++;
}
```

ลูป (Loop) ช่วยลดความซ้ำซ้อน
ประหยัดเวลา และทำให้โค้ดเป็นระเบียบ

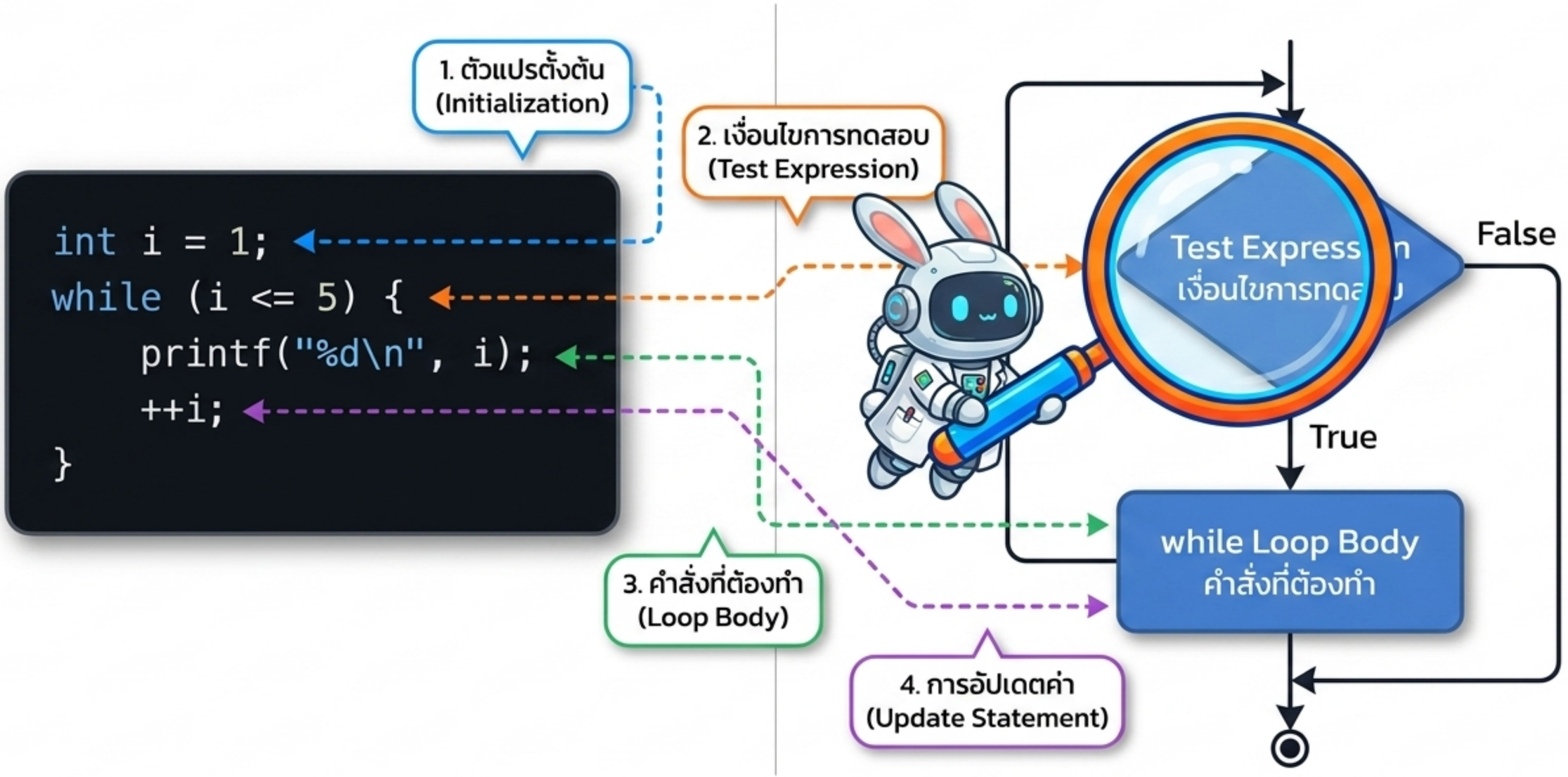
while loop คืออะไร?



ตราบใดที่เงื่อนไขเป็นจริง (True)
= โปรแกรมจะวนลูปทำงานต่อไปเรื่อยๆ

ถ้าเงื่อนไขเป็นเท็จ (False)
= โปรแกรมจะหยุดและออกจากลูปทันที

ผ่าโครงสร้าง while loop (Anatomy & Flowchart)



แกะรอยการทำงานแบบ Step-by-Step (Execution Trace)

รอบที่ (Iteration)	ค่าของ i	ตรวจสอบ i <= 3 (Condition)	ผลลัพธ์ (Output)	อัปเดต (Update)
รอบที่ 1	i = 1	1 <= 3 (True)	พิมพ์ 1	i กลายเป็น 2
รอบที่ 2	i = 2	2 <= 3 (True)	พิมพ์ 2	i กลายเป็น 3
รอบที่ 3	i = 3	3 <= 3 (True)	พิมพ์ 3	i กลายเป็น 4
รอบที่ 4	i = 4	4 <= 3 (False)	ออกจากลูป	-

Console Output

```
> 1
> 2
> 3
```

ตัวอย่างจริง: การนับเลขและหาผลรวมสะสม

Code Snippet

```c
int count = 1;
int sum = 0;
while (count <= 10) {
    sum += count;
    count++;
}
```

รูปแบบการบวกสะสม (Accumulator Pattern) ใช้บ่อยในระบบธุรกิจ เช่น การรวมยอดขายรายวัน

อย่าลืมอัปเดตค่าตัวแปรทุกครั้งเพื่อป้องกันลูปค้าง!

Terminal

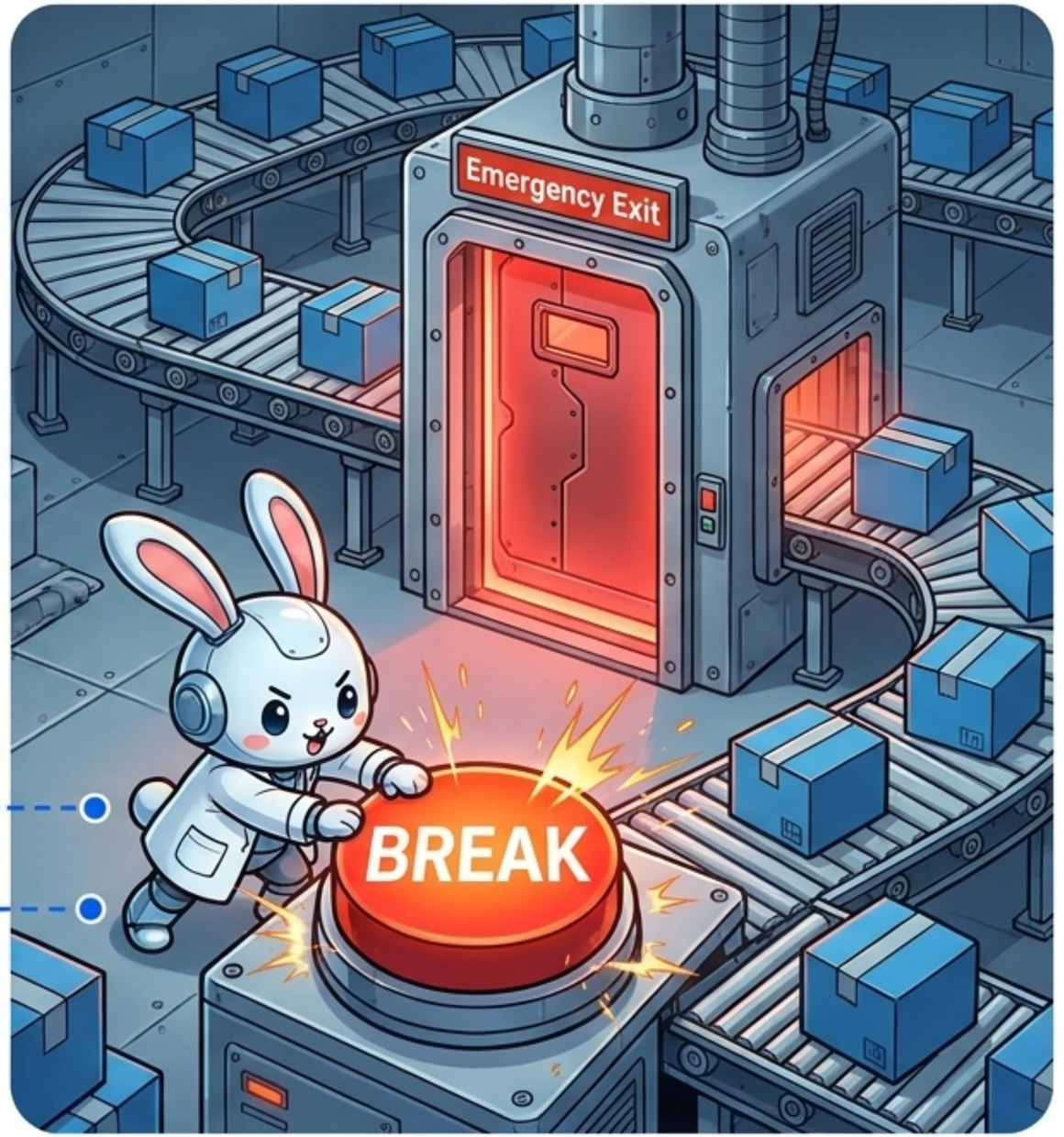
```
Sum = 55
```

เครื่องมือควบคุม #1:
คำสั่ง break (ปุ่มหยุดฉุกเฉิน)

```
1  if (found == 1) {
2      break;
3  }
```

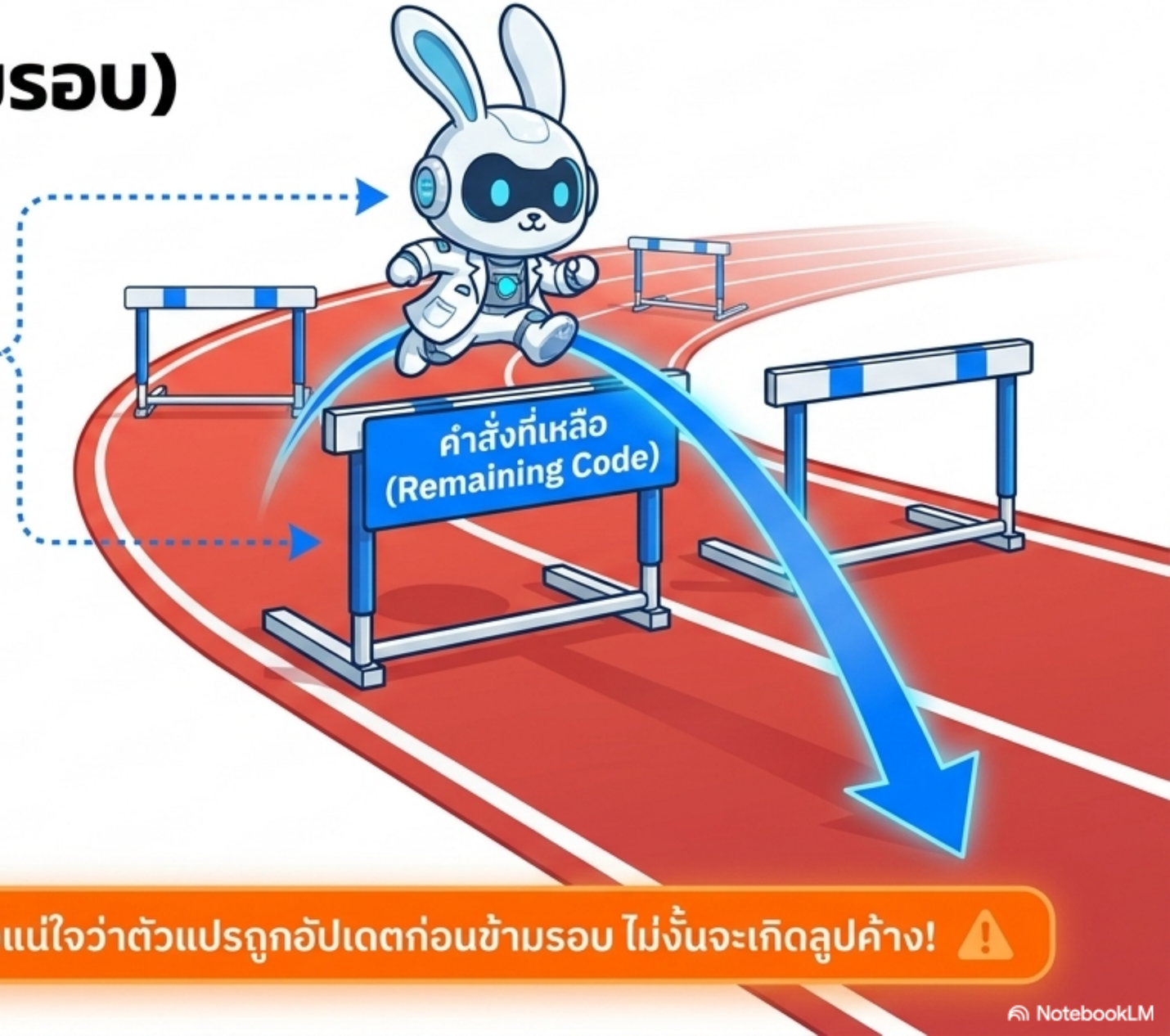
Concept: กระโดดออกจากลูป
ทันที โดยไม่ต้องรอให้เงื่อนไขหลักเป็นเท็จ

Best Use Case: ใช้เมื่อ ‘เจอสิ่งที่ค้นหาแล้ว’
(เช่น ค้นหารายชื่อในฐานข้อมูลเจอแล้ว
ไม่ต้องเสียเวลาหาต่อ)



เครื่องมือควบคุม #2: คำสั่ง continue (ปุ่มข้ามรอบ)

```
1   if (n % 2 != 0) {
2       continue;
3   }
```



Concept:
ข้ามคำสั่งที่เหลือในรอบปัจจุบัน
และกระโดดกลับไปเริ่มรอบใหม่ทันที

⚠ ระวัง! หากใช้ continue ใน while loop ต้องแน่ใจว่าตัวแปรถูกอัปเดตก่อนข้ามรอบ ไม่งั้นจะเกิดลูปค้าง! ⚠

The Danger Zone: ระวัง! Infinite Loop (ลูปอนันต์)

Danger Zone Checklist

ลืมอัปเดตตัวแปร
ตัวอย่างเช่น ลืมเขียน i++ ทำให้เงื่อนไขไม่เปลี่ยนและติดลูป

ตั้งเงื่อนไขผิด
เช่น while(1) หรือเงื่อนไขที่เป็น True เสมอโดยไม่ตั้งใจ

ใช้ continue ผิดที่
ทำให้โปรแกรมข้ามบรรทัดอัปเดตค่าไปอย่างถาวร

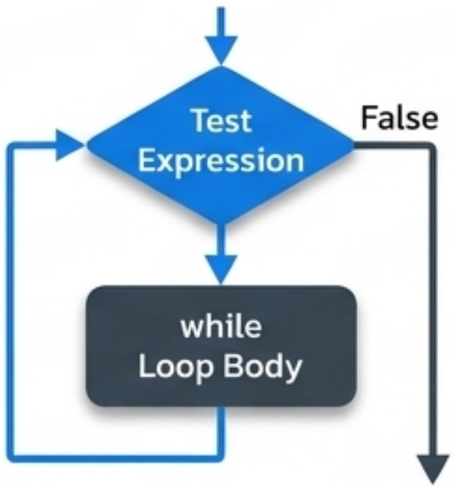
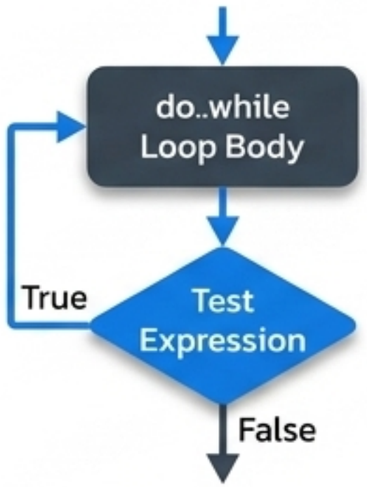
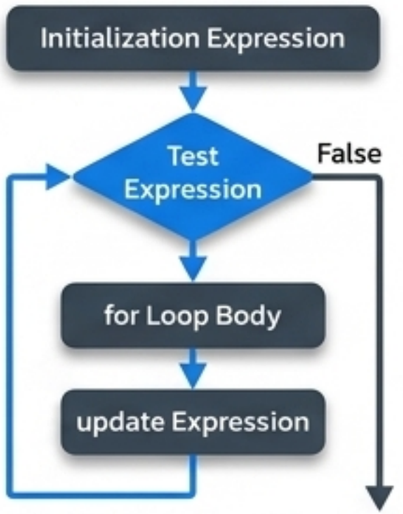
Solution: Safety Checklist – ตรวจสอบตัวอัปเดต (Updater) และทางออกฉุกเฉิน (Break) ทุกครั้งที่เขียนลูป!

Nested While Loops (ลูปซ้อนลูป)

การทำงานซ้อนกัน:
Loop นอกทำงาน 1 ครั้ง = Loop ในทำงานจนครบทุกรอบของตัวเอง

Real-World Application
เหมาะสำหรับการสร้างระบบที่มีแกน X และ Y เช่น:
- การวาดตารางหมากรุก
- การประมวลผลภาพ (Image Processing)
- ฐานข้อมูลแบบ Matrix

เปรียบเทียบลูป (The Ultimate Loop Cheat Sheet)

	while loop	do-while loop	for loop
			
เช็คเงื่อนไข	ตรวจสอบก่อนทำ (Entry-controlled)	ทำก่อนแล้วค่อยเช็ค (Exit-controlled)	ตรวจสอบก่อนทำ พร้อมอัปเดตในตัว
จำนวนรอบขั้นต่ำ	0 รอบ (อาจไม่ทำงานเลย)	1 รอบเสมอ	0 รอบ
เหมาะกับ	งานที่ไม่รู้จำนวนรอบชัดเจน	ระบบเมนูหรือการรอรับค่าจากผู้ใช้	งานที่รู้จำนวนรอบที่แน่นอน

สรุปใจความสำคัญ (Wrap Up)

- **3 องค์ประกอบสำคัญ:**
  ตัวเริ่มต้น -> เงื่อนไข ->
  การอัปเดตค่า (อย่าลืมเด็ดขาด!)

- **เครื่องมือควบคุม:**
  ใช้ **break** เพื่อหยุดฉุกเฉิน
  และ **continue** เพื่อข้ามรอบ

- **อันตราย:** ระวัง **Infinite Loop**
  ตรวจสอบเงื่อนไขการ
  จบการทำงานเสมอ

ผู้สอน: นายอัจฉริยะ มูลจันดา (วิทยาลัยการอาชีพขุนหาญ) สงสัยตรงไหน ทบทวนโค้ด หรือสอบถามอาจารย์ได้เลยครับ!