

# แนะนำการใช้ for loop ใน Python

Shutchon Premchaisawatt

ในภาษา Python for loop เป็นโครงสร้างการควบคุมการไหลของโปรแกรมที่ใช้ในการวนลูปหรือทำซ้ำชุดคำสั่งเป็นจำนวนครั้งที่กำหนด For loop เป็นหนึ่งในโครงสร้างการควบคุมการไหลที่สำคัญที่สุดใน Python และใช้กันอย่างแพร่หลายในโปรแกรมต่างๆ การใช้ for loop ช่วยให้คุณสามารถลดจำนวนบรรทัดโค้ดและเพิ่มประสิทธิภาพของโปรแกรม



```
In [10]: my_string = "python"
x = 0

for i in my_string:
    x = x + 1
    print(my_string[0:x])

for i in my_string:
    x = x - 1
    print(my_string[0:x])

p
py
pyt
pyth
pytho
python
pytho
pyth
pyt
PY
p
```

# ความสำคัญของการใช้ for loop

1

## ลดจำนวนบรรทัดโค้ด

การใช้ for loop ช่วยลดจำนวนบรรทัดโค้ดที่จำเป็นในการทำซ้ำชุดคำสั่ง ทำให้โค้ดของคุณอ่านง่าย  
ง่ายขึ้นและบำรุงรักษาง่ายขึ้น

2

## เพิ่มประสิทธิภาพของโปรแกรม

การใช้ for loop ช่วยลดจำนวนครั้งที่โปรแกรมต้องประมวลผลคำสั่ง ซึ่งช่วยเพิ่มประสิทธิภาพของ  
โปรแกรมและทำให้รันเร็วขึ้น

3

## เขียนโค้ดที่ชัดเจน

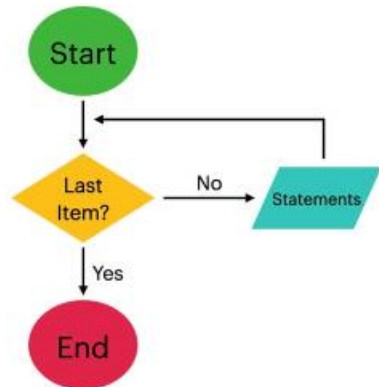
การใช้ for loop ช่วยให้โค้ดของคุณอ่านง่ายขึ้นและเข้าใจง่ายขึ้น เพราะการทำซ้ำชุดคำสั่งถูกจัด  
กลุ่มอย่างชัดเจน

4

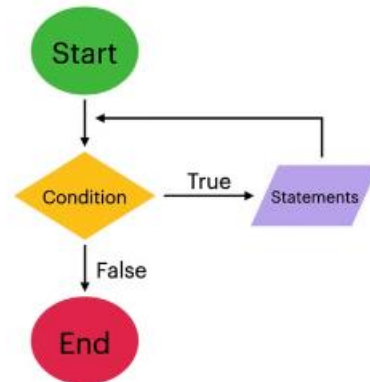
## ปรับใช้ได้ง่าย

for loop สามารถปรับใช้ได้ง่ายในกรณีที่ต้องการทำซ้ำชุดคำสั่งเป็นจำนวนครั้งที่แตกต่างกัน

For Loop



While Loop



# การใช้ for loop แบบง่าย

ในการใช้ for loop ใน Python เราจะใช้คำสั่ง for ตามด้วยตัวแปรวนลูป คำสั่ง in และ iterable object ตัวแปรวนลูปจะรับค่าจาก iterable object ในแต่ละรอบของการวนลูป ตัวอย่างเช่น:

```
>>> fruits = ["apple", "banana", "cherry"]
>>> for fruit in fruits:
...     print(fruit)
...
apple
banana
cherry
```

ในตัวอย่างนี้ fruits คือ iterable object ซึ่งเป็น list ของชื่อผลไม้ ตัวแปร fruit จะรับค่าจาก list นี้ในแต่ละรอบของการวนลูป ซึ่งจะทำให้คำสั่ง print(fruit) แสดงผลชื่อผลไม้ตามลำดับใน list นี้



Python for loop

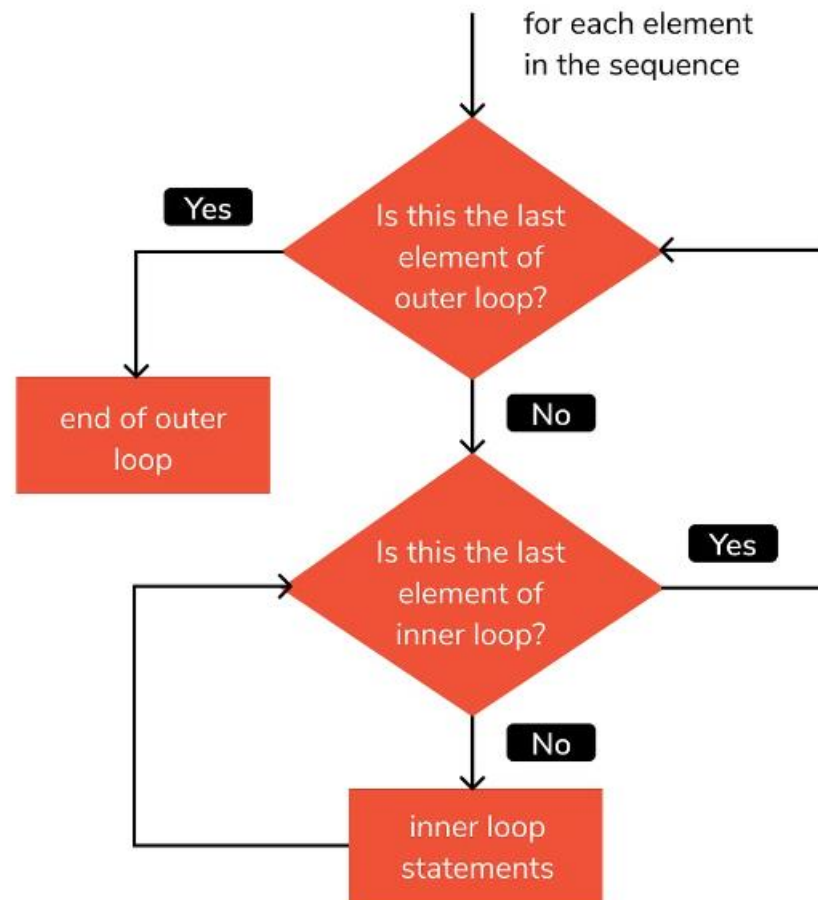
# การใช้ for loop ซ้อนกัน

การใช้ for loop ซ้อนกัน เป็นการใช้ for loop ภายใน for loop อีกครั้ง ซึ่งจะทำให้โปรแกรมสามารถ  
สามารถทำซ้ำชุดคำสั่งได้หลายชั้น

ตัวอย่างเช่น:

```
>>> for i in range(3):  
...     for j in range(2):  
...         print(i, j)  
...  
0 0  
0 1  
1 0  
1 1  
2 0  
2 1
```

ในตัวอย่างนี้ for loop แรกวนลูป 3 รอบ ในแต่ละรอบ for loop ที่สองจะวนลูป 2 รอบ ทำให้ผลลัพธ์  
แสดงค่าของ i และ j ในแต่ละรอบของการวนลูป





# ตัวอย่างการใช้ for loop ซ้อนกัน

ตัวอย่างการใช้ for loop ซ้อนกันเพื่อแสดงตารางการคูณ:

```
>>> for i in range(1, 11):
...     for j in range(1, 11):
...         print(f"{i} x {j} = {i * j}", end="\t")
...     print()
...
```

1 x 1 = 1 1 x 2 = 2 1 x 3 = 3 1 x 4 = 4 1 x 5 = 5 1 x 6 = 6 1 x 7 = 7 1 x 8 = 8 1  
x 9 = 9 1 x 10 = 10

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

3 x 1 = 3 3 x 2 = 6 3 x 3 = 9 3 x 4 = 12 3 x 5 = 15 3 x 6 = 18  
3 x 7 = 21 3 x 8 = 24 3 x 9 = 27 3 x 10 = 30

4 x 1 = 4 4 x 2 = 8 4 x 3 = 12 4 x 4 = 16 4 x 5 = 20 4  
x 6 = 24 4 x 7 = 28 4 x 8 = 32 4 x 9 = 36 4 x 10 = 40

5 x 1 = 5 5 x 2 = 10 5 x 3 = 15 5 x 4 = 20 5 x 5 = 25  
5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45  
5 x 10 = 50

6 x 1 = 6 6 x 2 = 12 6 x 3 = 18 6 x 4 = 24 6 x 5 = 30  
6 x 6 = 36 6 x 7 = 42 6 x 8 = 48 6 x 9 = 54  
6 x 10 = 60

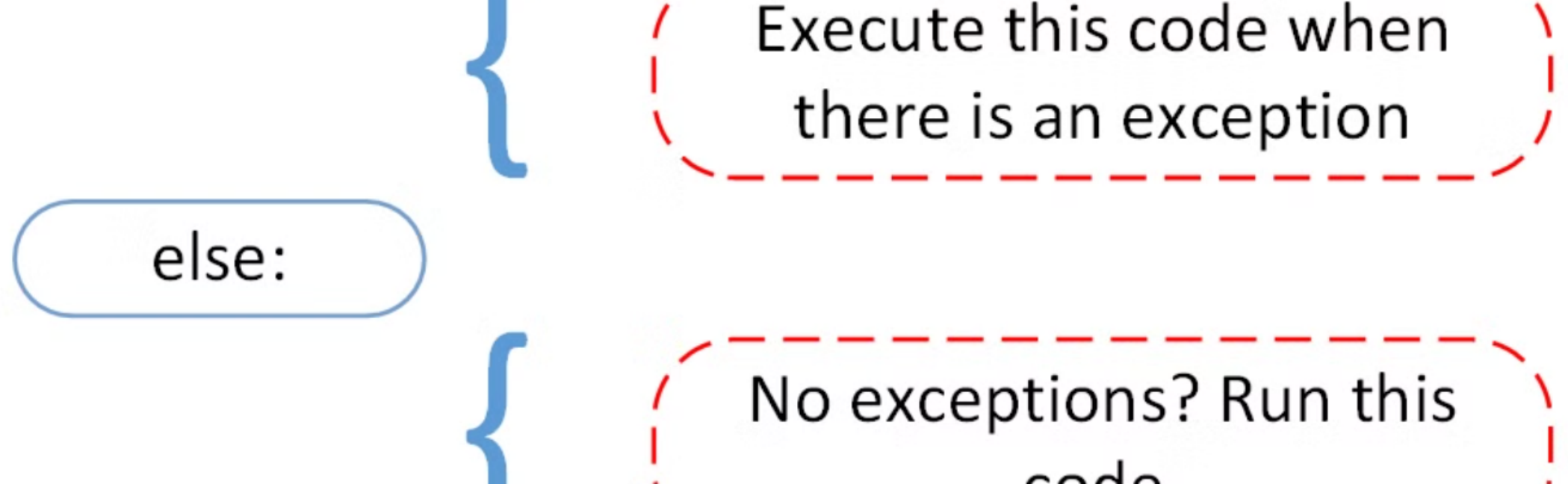
7 x 1 = 7 7 x 2 = 14 7 x 3 = 21 7 x 4 = 28 7 x 5 = 35  
7 x 6 = 42 7 x 7 = 49 7 x 8 = 56 7 x 9 = 63  
7 x 10 = 70

## Times Tables 1 to 12

1 times table	2 times table	3 times table	4 times table
1 x 1 = 1	1 x 2 = 2	1 x 3 = 3	1 x 4 = 4
2 x 1 = 2	2 x 2 = 4	2 x 3 = 6	2 x 4 = 8
3 x 1 = 3	3 x 2 = 6	3 x 3 = 9	3 x 4 = 12
4 x 1 = 4	4 x 2 = 8	4 x 3 = 12	4 x 4 = 16
5 x 1 = 5	5 x 2 = 10	5 x 3 = 15	5 x 4 = 20
6 x 1 = 6	6 x 2 = 12	6 x 3 = 18	6 x 4 = 24
7 x 1 = 7	7 x 2 = 14	7 x 3 = 21	7 x 4 = 28
8 x 1 = 8	8 x 2 = 16	8 x 3 = 24	8 x 4 = 32
9 x 1 = 9	9 x 2 = 18	9 x 3 = 27	9 x 4 = 36
10 x 1 = 10	10 x 2 = 20	10 x 3 = 30	10 x 4 = 40
11 x 1 = 11	11 x 2 = 22	11 x 3 = 33	11 x 4 = 44
12 x 1 = 12	12 x 2 = 24	12 x 3 = 36	12 x 4 = 48

5 times table	6 times table	7 times table	8 times table
1 x 5 = 5	1 x 6 = 6	1 x 7 = 7	1 x 8 = 8
2 x 5 = 10	2 x 6 = 12	2 x 7 = 14	2 x 8 = 16
3 x 5 = 15	3 x 6 = 18	3 x 7 = 21	3 x 8 = 24
4 x 5 = 20	4 x 6 = 24	4 x 7 = 28	4 x 8 = 32
5 x 5 = 25	5 x 6 = 30	5 x 7 = 35	5 x 8 = 40
6 x 5 = 30	6 x 6 = 36	6 x 7 = 42	6 x 8 = 48
7 x 5 = 35	7 x 6 = 42	7 x 7 = 49	7 x 8 = 56
8 x 5 = 40	8 x 6 = 48	8 x 7 = 56	8 x 8 = 64
9 x 5 = 45	9 x 6 = 54	9 x 7 = 63	9 x 8 = 72
10 x 5 = 50	10 x 6 = 60	10 x 7 = 70	10 x 8 = 80
11 x 5 = 55	11 x 6 = 66	11 x 7 = 77	11 x 8 = 88
12 x 5 = 60	12 x 6 = 72	12 x 7 = 84	12 x 8 = 96

9 times table	10 times table	11 times table	12 times table
1 x 9 = 9	1 x 10 = 10	1 x 11 = 11	1 x 12 = 12
2 x 9 = 18	2 x 10 = 20	2 x 11 = 22	2 x 12 = 24
3 x 9 = 27	3 x 10 = 30	3 x 11 = 33	3 x 12 = 36
4 x 9 = 36	4 x 10 = 40	4 x 11 = 44	4 x 12 = 48
5 x 9 = 45	5 x 10 = 50	5 x 11 = 55	5 x 12 = 60
6 x 9 = 54	6 x 10 = 60	6 x 11 = 66	6 x 12 = 72
7 x 9 = 63	7 x 10 = 70	7 x 11 = 77	7 x 12 = 84
8 x 9 = 72	8 x 10 = 80	8 x 11 = 88	8 x 12 = 96
9 x 9 = 81	9 x 10 = 90	9 x 11 = 99	9 x 12 = 108
10 x 9 = 90	10 x 10 = 100	10 x 11 = 110	10 x 12 = 120
11 x 9 = 99	11 x 10 = 110	11 x 11 = 121	11 x 12 = 132
12 x 9 = 108	12 x 10 = 120	12 x 11 = 132	12 x 12 = 144



## ข้อควรระวังในการใช้ for loop ซ้อนกัน

การใช้ for loop ซ้อนกันเป็นเทคนิคที่ทรงพลัง แต่ก็มีข้อควรระวังบางประการ

ในกรณีที่จำนวนรอบของการวนลูปมากเกินไป อาจส่งผลให้โปรแกรมทำงานช้าลง หรือใช้ทรัพยากรมากขึ้น

นอกจากนี้ การใช้ for loop ซ้อนกันในบางกรณีอาจทำให้โค้ดอ่านยากขึ้นและบำรุงรักษายากขึ้น ควรพิจารณาโครงสร้างข้อมูลที่เหมาะสมและแนวทางการเขียนโค้ดที่ดีเพื่อหลีกเลี่ยงข้อผิดพลาดและเพิ่มประสิทธิภาพของโปรแกรม

# การใช้ for loop ร่วมกับ if-else

การใช้ for loop ร่วมกับ if-else เป็นการตรวจสอบเงื่อนไขในแต่ละรอบของการวนลูป เพื่อดำเนินการตามเงื่อนไขที่กำหนด

ตัวอย่างเช่น:

```
>>> numbers = [1, 2, 3, 4, 5]
>>> for number in numbers:
...     if number % 2 == 0:
...         print(f"{number} is even")
...     else:
...         print(f"{number} is odd")
...
1 is odd
2 is even
3 is odd
4 is even
5 is odd
```

ในตัวอย่างนี้ for loop จะวนลูปผ่าน list ของ numbers และใช้ if-else เพื่อตรวจสอบว่าแต่ละหมายเลขเป็นเลขคู่หรือเลขคี่

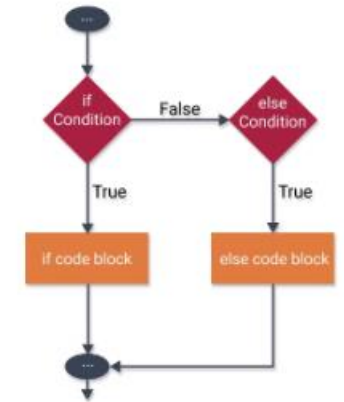
## Python if elif else Statement

### if statement:

Used to conditionally execute a block of code

#### Syntax:

```
if condition:
    # statement
    ...
```

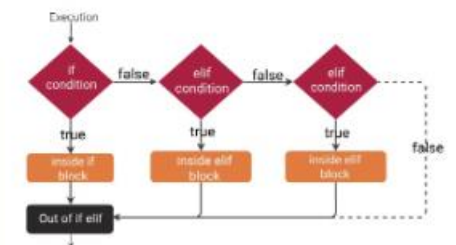


### elif statement:

Used to check other condition when condition of if statement is false

#### Syntax:

```
if condition:
    # statement
    ...
elif condition2:
    # statement
    ...
elif condition3:
    # statement
    ...
```

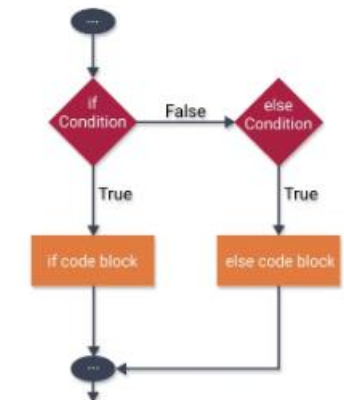


### else statement:

Else block is finally executed when all other conditions are false for an if statement

#### Syntax:

```
if condition:
    # statement
    ...
elif condition2:
    # statement
    ...
else:
    # statement
    ...
```



### ternary operator:

One line if-else block that returns a value based on condition

#### Syntax:

```
value1 if condition else value2
```



# การใช้ for loop กับ list และ dictionary

For loop สามารถใช้กับ list และ dictionary เพื่อเข้าถึงข้อมูลในแต่ละองค์ประกอบ

ตัวอย่างเช่น:

```
>>> my_list = ["apple", "banana", "cherry"]
>>> for item in my_list:
...     print(item)
...
apple
banana
cherry
```

```
>>> my_dict = {"name": "John", "age": 30, "city": "New York"}
>>> for key, value in my_dict.items():
...     print(f"{key}: {value}")
...
name: John
age: 30
city: New York
```

ในตัวอย่างแรก for loop จะวนลูปผ่าน list ของ my\_list และแสดงผลค่าในแต่ละองค์ประกอบของ list ในตัวอย่างที่สอง for loop จะวนลูปผ่าน dictionary ของ my\_dict และแสดงผล key และ value ของแต่ละองค์ประกอบใน dictionary



# การใช้ for loop กับ range()

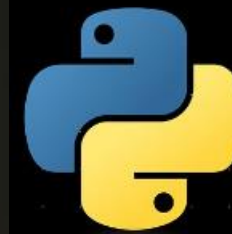
range() เป็นฟังก์ชันใน Python ที่สร้างลำดับของจำนวนเต็ม ซึ่งสามารถใช้กับ for loop เพื่อวนลูปตามจำนวนครั้งที่กำหนด

ตัวอย่างเช่น:

```
>>> for i in range(5):  
...     print(i)  
...  
0  
1  
2  
3  
4
```

ในตัวอย่างนี้ for loop จะวนลูป 5 รอบ เพราะ range(5) สร้างลำดับของจำนวนเต็มตั้งแต่ 0 ถึง 4

## PYTHON RANGE() FUNCTION



range(stop)

range(start, stop)

range(start, stop, step\_size)

# สรุปและแนะนำแนวทางการใช้ for loop ที่เหมาะสม

For loop เป็นโครงสร้างการควบคุมการไหลที่สำคัญใน Python และสามารถใช้ในหลากหลายกรณี

การเลือกใช้ for loop แบบง่ายหรือ for loop ซ้อนกันขึ้นอยู่กับความซับซ้อนของปัญหาและโครงสร้างข้อมูลที่ใช้

ควรพิจารณาแนวทางการเขียนโค้ดที่ดีเพื่อให้โค้ดอ่านง่ายและบำรุงรักษาง่าย

นอกจากนี้ ควรเลือกใช้ for loop ในกรณีที่ต้องการวนลูปชุดคำสั่งเป็นจำนวนครั้งที่กำหนด หรือวนลูปผ่านโครงสร้างข้อมูลอย่าง list หรือ dictionary

