

คำสั่งทำงานซ้ำ(Loops)

ประเภทของคำสั่งทำงานซ้ำ

- While loop
- For loop
- Nested loops

While loop

- while loop เป็นคำสั่งในการทำงานซ้ำในภาษา Python ที่จะทำงานซ้ำตราบเท่าที่เงื่อนไขยังคงเป็นจริง (True) โดยจะหยุดการทำงานเมื่อเงื่อนไขเป็นเท็จ (False)

รูปแบบ

`while` condition:

code block

statements

โดยที่:

- condition คือเงื่อนไขที่เป็น Boolean expression ซึ่งจะถูกประเมินในแต่ละรอบของการทำงานซ้ำ
- code block คือส่วนของคำสั่งที่จะถูกทำซ้ำตราบเท่าที่เงื่อนไขยังเป็นจริง



```
count = 1
```

```
while count <= 5:
```

```
    print(count)
```

```
    count += 1
```



1

2

3

4

5

รับข้อความจากผู้ใช้จนกว่าจะพิมพ์คำว่า "exit"

```
user_input = ""
```

```
while user_input != "exit":
```

```
    user_input = input("Enter some text (or 'exit' to quit): ")
```

```
    if user_input != "exit":
```

```
        print("You entered:", user_input)
```


หาผลรวมของเลขจนกว่าผลรวมจะมากกว่า 100

```
total = 0
```

```
number = 0
```

```
while total <= 100:
```

```
    number += 1
```

```
    total += number
```

```
    print(f"Current total: {total}")
```

```
print(f"The sum exceeded 100 after adding {number}")
```

คำสั่ง while-else

ใน Python เราสามารถใช้คำสั่ง else ร่วมกับ while loop ได้ คำสั่ง else จะถูกประมวลผลหลังจากที่ loop สิ้นสุดลง โดยที่เงื่อนไขของ while loop เป็นเท็จ



while condition:

code block

else:

final code block

หาค่าพหุนามของตัวเลขจนกว่าผลลัพธ์จะมากกว่า 500

```
result = 0
```

```
base = 1
```

```
while result < 500:
```


```
    result += base ** 2
```

```
    base += 1
```

```
    print(f"Current result: {result}")
```

```
else:
```

```
    print(f"The result first exceeded 500 when base was {base}")
```



```
i = 1
```

```
while i < 6:
```

```
    print(i)
```

```
    i += 1
```

```
else:
```

```
    print("i is no longer less than 6")
```

เราสามารถใช้คำสั่ง **break** ร่วมกับ **while loop**

- เพื่อหยุดการทำงานของลูปก่อนที่จะครบรอบตามเงื่อนไขที่กำหนดไว้
- เมื่อคำสั่ง **break** ถูกเรียกใช้ ควบคุมการทำงานจะออกจาก **while loop** ทันที และไปทำงานในบรรทัดต่อไปภายนอกลูป



while condition:

code block

if another_condition:

break

remaining code block

- เมื่อ `another_condition` เป็นจริง คำสั่ง `break` จะถูกเรียกใช้ และการทำงานจะออกจาก `while loop` ทันที

รับข้อความจากผู้ใช้งาน และหยุดเมื่อผู้ใช้พิมพ์คำว่า "exit"

```
while True:
```

```
    user_input = input("Enter some text ('exit' to quit): ")
```

```
    if user_input == "exit":
```

```
        break
```

```
    print("You entered:", user_input)
```

คำสั่ง `continue` ร่วมกับ `while loop`

- คำสั่งนี้จะข้ามการทำงานที่เหลือในรอบปัจจุบันของกลุ่ม และไปทำในรอบถัดไปทันที



while condition:

code block

if another_condition:

 continue

remaining code block

- เมื่อ `another_condition` เป็นจริง คำสั่ง `continue` จะถูกเรียกใช้ ควบคุมการทำงานจะข้ามส่วนที่เหลือของรอบปัจจุบัน และไปทำในรอบถัดไปของ `while loop` ทันที

แสดงเลขคี่ในช่วง 1-10

```
num = 0
```

```
while num < 10:
```

```
    num += 1
```

```
    if num % 2 == 0:
```

```
        continue
```

```
    print(num)
```

รับข้อความจากผู้ใช้ จนกว่าจะมีข้อความที่ไม่ว่างเปล่า

```
while True:
```

```
    user_input = input("Enter some text (or just 'Enter' to quit): ").strip()
```

```
    if not user_input:
```

```
        break
```

```
    if user_input.isdigit():
```

```
        continue
```

```
    print("You entered:", user_input)
```

Single statement while Block

- คือการเขียน while loop ในบรรทัดเดียวกัน ซึ่ง เป็นรูปแบบการเขียนที่สั้นและกะทัดรัด สามารถ นำมาใช้ได้เมื่อมีคำสั่งเพียงบรรทัดเดียวภายใต้ while loop



`while condition: statement`

`count = 1`

`while count <= 5: print(count); count += 1`

คำสั่ง for Loop

- คำสั่ง for loop ในภาษา Python ใช้สำหรับวนลูปผ่านสมาชิกของอ็อบเจกต์ที่สามารถนำมาเรียกค่าได้ (iterable) เช่น ลิสต์ (list) ทูเพิล (tuple) เรนจ์ (range) ข้อความ (string) เป็นต้น

for **item** in **iterable**:
code block

โดยที่:

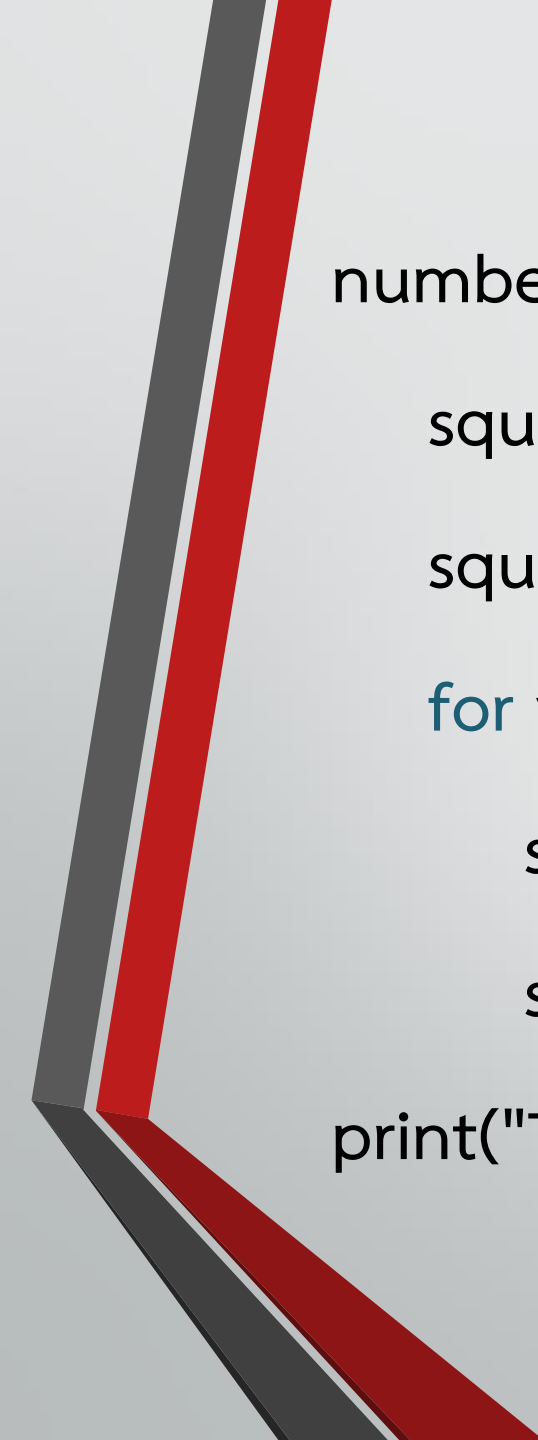
- **item** คือตัวแปรที่จะถูกกำหนดค่าเป็นสมาชิกของ **iterable** ในแต่ละรอบของลูป
- **iterable** คืออ็อบเจกต์ที่สามารถนำมาวนลูปได้ เช่น **ลิสต์** **ทูเพิล** **เรนจ์** หรือ **ข้อความ**

วนลูปผ่านสมาชิกของลิสต์

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

```
for fruit in fruits:
```

```
    print(fruit)
```



```
numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]
```

```
square = 0
```

```
squares = []
```

```
for value in numbers:
```

```
    square = value ** 2
```

```
    squares.append(square)
```

```
print("The list of squares is", squares)
```

คำสั่ง `else` ร่วมกับ `for loop`

- คำสั่ง `else` จะถูกประมวลผลหลังจากที่ `loop` ผ่านสมาชิกทั้งหมดของอ็อบเจกต์ที่ระบุแล้ว

```
tuple_ = (3, 4, 6, 8, 9, 2, 3, 8, 9, 7)
```

```
for value in tuple_:
```

```
    if value % 2 != 0:
```

```
        print(value)
```

```
    else:
```

```
        print("These are the odd numbers present in  
the tuple")
```

ตรวจสอบว่ามีเลขคู่ในลิสต์หรือไม่

```
numbers = [1, 3, 5, 7, 9]
```

```
for num in numbers:
```

```
    if num % 2 == 0:
```

```
        print(f"Found an even number: {num}")
```

```
        break
```

```
else:
```

```
    print("No even numbers found.")
```

หาผลรวมของเลขคู่ในลิสต์ หากไม่มีเลขคู่จะแสดงข้อความ

```
numbers = [1, 3, 5, 6, 9, 8]
```

```
total = 0
```

```
for num in numbers:
```

```
    if num % 2 == 0:
```

```
        total += num
```

```
else:
```

```
    if total == 0:
```

```
        print("No even numbers found.")
```

```
else:
```

```
    print(f"Sum of even numbers: {total}")
```


ฟังก์ชัน range() ร่วมกับคำสั่ง for loop

ฟังก์ชัน range() ร่วมกับคำสั่ง for loop เพื่อวน
ลูปผ่านช่วงของตัวเลขได้ range() จะสร้างอนุกรม
เลขจำนวนเต็มภายในช่วงที่กำหนด

1.range(stop)

2.range(start, stop)

3.range(start, stop, step)

โดยที่:

- start คือจุดเริ่มต้นของอนุกรม (ค่าเริ่มต้นคือ 0 ถ้าไม่ระบุ)
- stop คือจุดสิ้นสุดของอนุกรม (ค่าสุดท้ายจะไม่ถูกรวมอยู่ในอนุกรม)
- step คือค่าก้าวของอนุกรม (ค่าเริ่มต้นคือ 1 ถ้าไม่ระบุ)

วนลูปตัวเลขจาก 0 ถึง 4

```
for i in range(5):
```

```
    print(i)
```

```
0
```

```
1
```

```
2
```

```
3
```

```
4
```

วนลูปตัวเลขจาก 3 ถึง 7

```
for i in range(3, 8):  
    print(i)
```

```
3  
4  
5  
6  
7
```

วนลูปตัวเลขจาก 0 ถึง 10 โดยเพิ่มขึ้นทีละ 2

```
for i in range(0, 11, 2):  
    print(i)
```

```
0  
2  
4  
6  
8  
10
```



แบบฝึกหัด