Repetition Structures = การทำซ้ำ

Motivation : พิมพ์ตัวเลข 1-10 ต่อหนึ่งบรรทัด

①
```
print(1)
print(2)
print(3)
  ⋮
print(10)
```
เยอะ!! → ถ้าต้องทำ 1-70?

② ต้องรับค่าจาก user 10 ครั้ง
```
x1 = input(...)
  ⋮
x10 = input(...)
```

2 Repetition Structures

1) while : ทำซ้ำในขณะที่เงื่อนไขเป็นจริง
2) for : ทำซ้ำตามของที่ให้มาตามลำดับ

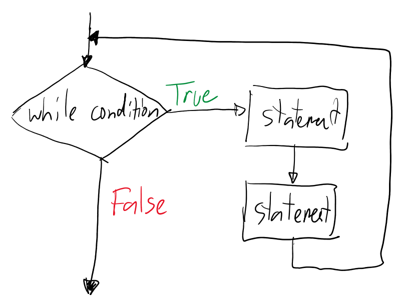
While Structure

- Condition loop: ทำจนกว่าเงื่อนไขเป็น False
- Syntax:

```
while condition:
    statement
    statement
    statement
```

ช่องว่าง บอกว่าอยู่ใน structure

① เช็คเงื่อนไข (True/False)
② ถ้า True เข้ามาทำ while-block
③ ทำจบ while-block วนมาเช็ค condition
④ False ออกจาก while loop.



EX

```
num = 0
while num < 3:
    print(num)
    num = num+1
print("Done")
```

True/False; True → while-block; False → print("Done")

Output:
0
1
2
Done

~~กาก~~ → เท่ากัน
```
num=0
print(num)
num = num+1
print(num)
num = num+1
print(num)
num = num+1
print("Done")
```

Trace:
```
num = 0
# loop 1
while 0 < 3:  → True ✓
    print(0)          → 0
    num = 0+1 → 1      [1] num
# loop 2
while 1 < 3   → True ✓
    print(1)          → 1
    num = 1+1 → 2      [2] num
# loop 3
while 2 < 3   → True ✓
    print(2)          → 2
    num = 2+1 → 3      [3] num
# loop 4
while 3 < 3:  → False ✗ ออกจาก loop
print("Done")         → Done
```

(✳) ถ้าเงื่อนไขไม่เป็น (False) เลย จะไม่หยุดวนลูป

(✳) Input Validation

↳ การเช็คข้อมูลนำเข้าว่าถูกต้องตามเงื่อนไข
↳ เช่น เช็คว่าอายุต้องมากกว่าเท่ากับ 0
↳ Idea: ถ้า user กรอกไม่ถูกเงื่อนไข → รับใหม่

↳ เช่น เช็คว่า อายุ ต้อง มากกว่า เท่ากับ 0

↳ Idea: ถ้า user ข้อมูลไม่ถูกเงื่อนไข → ถามใหม่ จนกว่าจะถูก

while loop:

Ex
```
age = int(input("Enter age:"))
while age < 0:
    print("Age must not be negative")
    age = int(input("Enter age:"))
print(f"Your age is {age}")
```
False

ถ้าออกจาก loop → age ถูกต้อง ตามเงื่อนไข

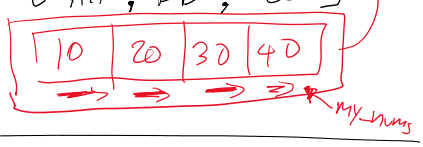
List → ข้อมูล หลายๆ ตัว เก็บรวมกัน

โดยการเข้าถึงข้อมูล ทำได้ด้วยการ for loop และ การเข้า index

Syntax: `list_name = [value1, value2, value3]`

e.x.
```
my_nums = [10, 20, 30, 40]
my_words = ['AA', "BB", "CC"]
```

* การเก็บข้อมูลในตาราง



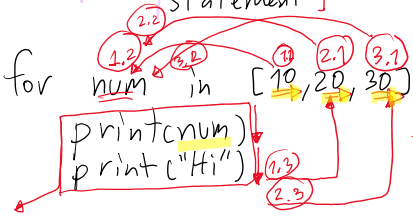
For loop

↳ loop เข้าไปใน list (sequence) ตามจำนวนไปของสมาชิกแต่ละตัว

Syntax:
```
for var in list:
    statement
    statement
    statement
```
var = ตัวแปรมาเก็บค่าของในlistตามลำดับ

for-block

Ex
```
for num in [10, 20, 30]:
    print(num)
    print("Hi")
```




```
# loop 1
num = 10
print(10)   → 10
print('Hi') → Hi
# loop 2
num = 20
print(20)   → 20
print('Hi') → Hi
# loop 3
num = 30
print(30)   → 30
print('Hi') → Hi
```

- **Calculate a sum of numbers**

↳ รับ list ของตัวเลขมา (เช่น [10,20,30])

↳ รับ list ของตัวเลขมา (เช่น [10,20,30])

หาผลรวม ของตัวเลขทั้งหมดใน list → 10+20+30 = 60

10	20	30

total ← + 10 +20 +30

(ทำซ้ำในแต่ละ loop)


```python
total = 0              # ← ตัวแปรมาเก็บค่าผลรวม
nums = [10, 20, 30]
for num in nums:
    total = total + num    # ] ทำซ้ำ
print(total)
```

total = total + num → total += num

total = 0
#loop 1
num = 10
total = 0 + 10 = 10
#loop2
num = 20
total = 10 + 20 = 30
#loop 3
num = 30
total = 30 + 30 = 60
print(60) → 60

Counting : นับของ

→ นับว่ามีของที่ต้องการอยู่ใน list กี่ตัว

→ เช่น นับเลขคู่ใน list

(% 2 == 0)

4 (+1)	6 (+1)	7 (+0)	9 (+0)
เช็คเลขคู่ ใช่ +1	เช็ค ใช่ +1	เช็ค ไม่ใช่	เช็ค ไม่ใช่

(*) ทำซ้ำ → เช็คเลขคู่ ถ้าใช่ +1 เข้าไปใน count (ตัวแปร)

```python
count = 0
nums = [4, 6, 7, 9]
for num in nums:
    if num % 2 == 0:       # ← เช็คแต่ละตัว  x ทำซ้ำ
        count = count + 1  # ← อัพเดต count
print(count)
```

count = count + 1 → count += 1

#loop 1
num = 4
if 4 % 2 == 0:
count = 0+1
#loop 2

Short-hand notation

count = count + 1

↓

count += 1

อัพเดตค่า

- x = x + 1 → x += 1
- y = y - 2 → y -= 2

- z = z * 3 ⟶ z *= 3
- m = m / 4 ⟶ m /= 4

Ex x = x + (10 * 2 - y) ⟶ x += (10 * 2 - y)

---

range() function

↳ function ที่สร้าง sequence ของตัวเลข (~ list)

↳ syntax: range(start, stop, step)

- start = ตัวแรก
- stop = จุดจบ ตัวสุดท้ายต้องมาก่อน stop (ตัวจบ คือ stop-1) ถ้า step เป็นบวก
- step = ค่าที่เพิ่ม เท่าไหร่ ไปในแต่ละตัว

Ex range(0, 5, 1) → [0, 1, 2, 3, 4]   (+1, +1, +1, +1)

range(10, 17, 2) → [10, 12, 14, 16]

range(10, 0, -3) → [10, 7, 4, 1]

*เลข 0 หยุด!

range แบบย่อ

syntax: 1) range(start, stop)

↳ step = 1 (fixed!!)

↳ ex range(1, 5) → [1, 2, 3, 4]

2) range(stop)

↳ start = 0 , step = 1 (fixed!!)

↳ e.x. range(4) → [0, 1, 2, 3]

---

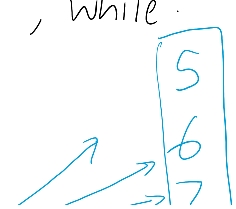
break statement

↳ เจอเมื่อไร ให้ออกจาก loop ทันที!!

↳ ใช้ได้ทั้ง for , while

Ex

```
num = 5
while True:
```



```
while True:
    print(num)
    if num > 7:
        break
    num = num + 1
```