Iteration

Ref Pseudocode

FOR, the counted loop

Iteration 迭代 repeats statements. Use a `FOR` loop when you know **before you start** how many times — a count-controlled 计数控制 loop.

```
FOR <identifier> ← <value1> TO <value2>
    <statements>
NEXT <identifier>
```

```
DECLARE Index : INTEGER
FOR Index ← 1 TO 5
    OUTPUT "Line ", Index
NEXT Index
```

The counter 计数器 starts at the first value, and the loop runs once for every value up to and **including** the second. `1 TO 5` runs five times.

STEP

`STEP` changes the size of each jump. A negative step counts down.

```
DECLARE Index : INTEGER
FOR Index ← 2 TO 10 STEP 2
    OUTPUT Index
NEXT Index
FOR Index ← 3 TO 1 STEP -1
    OUTPUT "Countdown ", Index
NEXT Index
```

Running totals

The commonest use: a variable outside the loop that the loop adds to.

```
DECLARE Index, Total : INTEGER
Total ← 0
FOR Index ← 1 TO 10
    Total ← Total + Index
NEXT Index
OUTPUT "Sum of 1 to 10 is ", Total
```


`Total ← 0` must sit **before** the loop. Inside, it would reset on every pass.

Common mistakes

- `NEXT` naming the wrong counter. Nested loops must close in the reverse order they opened: the inner `NEXT` first.
- Assuming `1 TO 5` runs four times. Both ends are included.
- Changing the counter inside the loop. Let `FOR` own it.

WHILE, the pre-condition loop

A `WHILE` loop tests **before** each pass, so it can run **zero** times. Use it when the number of repeats depends on something that happens while the loop runs.

```
WHILE <condition> DO
    <statements>
ENDWHILE
```

```
DECLARE Total : INTEGER
Total ← 1
WHILE Total < 100 DO
    Total ← Total * 2
ENDWHILE
OUTPUT "First power of two past 100: ", Total
```

Something inside the loop must eventually make the condition false, or the loop never ends — an **infinite loop** 无限循环.

```
DECLARE Count : INTEGER
Count ← 5
WHILE Count > 0 DO
    OUTPUT Count
    Count ← Count - 1
ENDWHILE
OUTPUT "Lift off"
```

Common mistakes

- Forgetting to change the variable the condition tests, giving an infinite loop.
- Leaving out `DO` or `ENDWHILE`.
- Using `WHILE` where the count is known. A `FOR` says what you mean in one line.

REPEAT, the post-condition loop

A `REPEAT` loop tests **after** each pass, so it always runs **at least once**.

```
REPEAT
    <statements>
UNTIL <condition>
```

That "at least once" is the reason to choose it: validating 验证 input, where you must ask before you can judge the answer.

```
DECLARE Number : INTEGER
Number ← 0
REPEAT
    Number ← Number + 25
    OUTPUT "Trying ", Number
UNTIL Number >= 100
OUTPUT "Reached ", Number
```

△ `UNTIL` states the condition that **stops** the loop; `WHILE` states the one that **keeps it going**. They are opposites, and swapping them is the classic error.

	Tests	Runs at least once?	Condition means
`FOR`	a counter	no (an empty range runs zero times)	—
`WHILE`	before	no	keep going while this is true
`REPEAT`	after	**yes**	stop when this becomes true

Common mistakes

- Writing `UNTIL Number < 100` when you mean `UNTIL Number >= 100`.
- Using `REPEAT` where the body must be able to run zero times.
- Forgetting that `UNTIL` ends the structure — there is no `ENDREPEAT`.