Bài 5: CẤU TRÚC LẶP VỚI SỐ LẦN BIẾT TRƯỚC FOR

I. Lặp

Trong quá trình thực hiện chương trình, ta sẽ gặp nhiều công việc được thực hiện lặp đi lặp lại. Các công việc đó có thể lặp lại giống nhau y hệt hoặc chỉ khác nhau về giá trị tính toán còn cách thức làm các công việc đó là giống nhau.

Có hai dạng lặp đó là:

- Lặp với số lần lặp biết trước.
- Lặp với số lần lặp chưa biết trước.

II. Lặp với số lần biết trước với câu lệnh for

1. Hàm range()

➢ Cú pháp:

range(stop)
range(start, stop)
range(start, stop, step)

➢ Giải thích cú pháp:

- start: giá trị bắt đầu (mặc định = 0).
- stop: giá trị kết thúc (không bao gồm stop).
- step: bước nhảy (mặc định = 1).

👉 Hàm range() tạo ra một dãy số nguyên theo quy luật đã cho.

➢ Chú ý:

- range(n) sẽ sinh ra dãy từ 0 đến n-1.
- step có thể âm để sinh dãy giảm dần.

➢ Ví dụ:

range(5) -> 0 1 2 3 4
range(2, 7) -> 2 3 4 5 6
range(1, 10, 2) -> 1 3 5 7 9

2. Câu lệnh for

❖ **Cú pháp chung:**

for <tên biến> **in** [tập các giá trị]:
    <khối lệnh>

Trong đó:

- <tên_biến>: biến lặp, nhận lần lượt từng giá trị trong tập giá trị.
- [tập_các_giá_trị]: có thể là range(), list, tuple, string, hoặc set.
- <khối_lệnh>: khối lệnh cần thực hiện lặp lại.

➢ Ví dụ:

✓ Ví dụ 1: Duyệt danh sách (list)

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

⇨ Kết quả:

```
apple
banana
cherry
```

✓ Ví dụ 2: Duyệt chuỗi (string)

```
for ch in "Python":
    print(ch, end=" ")
```

⇨ Kết quả:

P y t h o n

✓ Ví dụ 3: Duyệt tuple

```
numbers = (2, 4, 6, 8)
for n in numbers:
    print(n**2)
```

⇨ Kết quả:

4
16
36
64

✓ Ví dụ 4: Duyệt set

```
colors = {"red", "green", "blue"}
for c in colors:
    print(c)
```

⇨ Kết quả:

✓ Ví dụ 5: Duyệt với range()

```
for i in range(1, 6):
    print(i, end=" ")
```

⇨ Kết quả:

1 2 3 4 5

❖ **Cú pháp với vùng dữ liệu range()**

```
for <biến> in range(start, stop, step):
    < khối_lệnh>
```

➢ Giải thích cú pháp:

- biến: nhận lần lượt giá trị từ dãy số do range() sinh ra.
- khối_lệnh: phần lệnh được thực hiện lặp lại theo số lần.

👉 Lặp số lần biết trước (số vòng lặp phụ thuộc vào range()).

➢ Chú ý:

- Số lần lặp = độ dài của dãy range().
- Phải thụt đầu dòng khối lệnh bằng 4 khoảng trắng (hoặc tab).

➢ Ví dụ 1: In 5 lần câu chào

```
for i in range(5):
    print("Xin chào bạn")
```

👉 Kết quả:

Xin chào bạn
Xin chào bạn
Xin chào bạn
Xin chào bạn
Xin chào bạn

➢ Ví dụ 2: In các số từ 1 đến 5:

```
for i in range(1, 6):
    print(i) # in mỗi số trên 1 hàng
```

hoặc

```
for i in range(1, 6):
    print(i, end=" ") # in các số trên cùng 1 hàng cách nhau bởi khoảng trắng
```

➢ Ví dụ 3: In bình phương các số từ 1 đến 5

```
for i in range(1, 6):
```

```
    print(f"{i}^2 = {i**2}")
```

➢ Ví dụ 4: Tính tổng các số từ 1 đến 10

```
tong = 0
for i in range(1, 11):
   tong += i
print("Tổng từ 1 đến 10 là:", tong)
```

👉 Kết quả:

```
Tổng từ 1 đến 10 là: 55
```

❖ **Tóm tắt.**

Thành phần	Cú pháp	Ý nghĩa	Ví dụ minh họa
range(stop)	range(5)	Dãy số từ 0 → 4	0,1,2,3,4
range(start, stop)	range(2, 6)	Dãy số từ 2 → 5	2,3,4,5
range(start, stop, step)	range(1, 10, 2)	Dãy số từ 1 → 9, bước 2	1,3,5,7,9
for	for i in range(n): ...	Lặp n lần, i thay đổi theo range()	In ra 0,1,2,...,n-1

❖ **Câu lệnh break và câu lệnh continue.**

Câu lệnh break: break dùng để thoát hẳn khỏi vòng lặp (kể cả khi điều kiện vẫn còn đúng). Sau khi gặp break, chương trình sẽ dừng vòng lặp ngay lập tức và chạy tiếp các lệnh phía sau vòng lặp.

Câu lệnh continue: continue dùng để bỏ qua phần còn lại của vòng lặp hiện tại, và chuyển sang lần lặp tiếp theo. Điều kiện của vòng lặp vẫn được kiểm tra như bình thường.

➢ **So sánh:**

- break: dừng hẳn vòng lặp.
- continue: bỏ qua lần lặp hiện tại, tiếp tục vòng lặp mới.

❖ **Ta có thể kết hợp for với else:**

➢ Cú pháp:

```
for <tên_biến> in <tập_giá_trị>:
   <khối_lệnh_lặp>
else:
   <khối_lệnh_else>
```

➢ **Giải thích:**

- Vòng lặp for chạy hết dãy giá trị.
- Khi vòng lặp kết thúc bình thường (không bị break), thì khối lệnh trong else sẽ được thực hiện.
- Nếu vòng lặp bị dừng bởi break, thì khối else sẽ không được thực hiện.

➢ Ví dụ 1: Tìm số trong danh sách

```
numbers = [2, 4, 6, 8, 10]
x = 6
for n in numbers:
   if n == x:
      print(f"Tìm thấy {x} trong danh sách.")
      break
else:
```

```
    print(f"Không tìm thấy {x} trong danh sách.")
```

Giải thích:

- Vòng lặp for duyệt từng phần tử trong numbers.
- Nếu tìm thấy x, in ra kết quả và dừng vòng lặp bằng break → else không chạy.
- Nếu duyệt hết danh sách mà không gặp break, khối else sẽ chạy.

➢ Ví dụ 2: Kiểm tra số nguyên tố

```
n = 17
if n < 2:
    print(f"{n} không phải là số nguyên tố.")
else:
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            print(f"{n} không phải là số nguyên tố vì chia hết cho {i}.")
            break
    else:
        print(f"{n} là số nguyên tố.")
```

Giải thích:

* Nếu n < 2 thì không phải số nguyên tố.

* Vòng lặp for thử chia n cho các số từ 2 đến $\sqrt{n}$.

- Nếu tìm thấy số chia hết → dùng break → thoát vòng lặp, không chạy else.

- Nếu không có số nào chia hết → vòng lặp kết thúc bình thường → chạy else và kết luận n là số nguyên tố.

3. Các hàm toán học (math)

- Để sử dụng, cần import thư viện math (trừ các hàm abs, min, max và round).

Cú pháp: **import math**

hoặc **from math import * # (**dấu * để cho sử dụng tất cả các hàm trong thư viện math**)**

Ví dụ: tính căn bậc hai số học của 2 ta sử dụng hàm sqrt() trong thư viện math:

Cách 1: *import math*

print(math.sqrt(2))

Cách 2: from math import *

print(sqrt(2))

- Bảng các hàm toán học thường dùng:

Hàm	Chức năng	Ví dụ	Kết quả
math.sqrt(x)	Căn bậc hai của x	math.sqrt(9)	3.0
math.isqrt(x)	Lấy phần nguyên căn bậc hai của x	math.isqrt(9)	3
math.ceil(x)	Làm tròn lên	math.ceil(4.2)	5
math.floor(x)	Làm tròn xuống	math.floor(4.8)	4
math.pow(a, b)	Lũy thừa a^b	math.pow(2, 3)	8.0
math.exp(x)	Trả về giá trị của e mũ x (e**x)		
math.gcd(a, b)	Ước chung lớn nhất	math.gcd(12, 18)	6
math.lcm(a, b)	Bội chung nhỏ nhất (Python 3.9+)	math.lcm(4, 6)	12
math.factorial(n)	Giai thừa	math.factorial(5)	120

math.log(x, base)	Logarit cơ số base	math.log(8, 2)	3.0
math.pi	Hằng số π	math.pi	3.14159…
math.e	Hằng số e	math.e	2.71828…
math.trunc(x)	Tính phần nguyên của x(giống hàm int(x))	math.trunc(3.14)	3
math.sin(x)	Tính sin của góc có số đo x(rad)		
math.cos(x)	Tính cos của góc có số đo x(rad)		
math.tan(x)	Tính tan của góc có số đo x(rad)		
round(x,n)	Làm tròn số x đến n chữ số thập phân	round(3.14159,2)	3.14
abs(x)	Tính giá trị tuyệt đối của số x	abs(5)	5
max(a,b,c,…)	Trả về giá trị lớn nhất trong các số a, b, c, …	max(5,8,10)	10
min(a,b,c,…)	Trả về giá trị nhỏ nhất trong các số a, b, c, …	min(5,8,10)	5

III. BÀI TẬP.

Bài 1: In dãy số từ 1 đến 10.
Bài 2: In dãy số chẵn từ 2 đến 20.
Bài 3: In bình phương của các số từ 1 đến 10.
Bài 4: Cho số nguyên dương n. Hãy in ra các số tự nhiên nhỏ hơn n ($0 < n \leq 10^9$)
Bài 5: Tính tổng:
a, S = 1 + 2 + 3 + … + n (Tính tổng n số tự nhiên đầu tiên)
Cách 1: dùng vòng lặp For:
Cách 2: dùng công thức S = n(n+1)/2 (tối ưu hơn khi n là số lớn)
b, S = 1 + 3 + 5 + … + n (Tính tổng n số tự nhiên lẻ đầu tiên)
Cách 1: dùng vòng lặp For:
Cách 2: dùng công thức

```
# Số lượng số lẻ từ 1 đến n
k = (n + 1) // 2
S =  k * k   ( tối ưu hơn khi n là số lớn)
```

c, S = 2 + 4 + 6 + … + n (Tính tổng n số tự nhiên chẵn đầu tiên)
Cách 1: dùng vòng lặp For:
Cách 2: Thuật toán tối ưu
- Nếu n là lẻ → gán n = n - 1
- Áp dụng công thức: S = n × (n + 2) / 4
- In ra kết quả.

d, S = 1 - 2 + 3 – 4 + … n
Gợi ý: Dùng công thức
- Nếu n chẵn S = -n/2
- Nếu n lẻ S = (n+1)/2

e, S = -1 + 2 - 3 + 4 - … n
Gợi ý: Dùng công thức
- Nếu n chẵn S = n/2

- Nếu n lẻ S = -(n+1)/2

f, $S = 1^2 + 2^2 + 3^2 + \ldots + n^2$ (Tính tổng các bình phương của n số tự nhiên đầu tiên)

g, $S = 1^2 + 3^2 + 5^2 + \ldots + n^2$ (Tính tổng các bình phương của n số tự nhiên lẻ đầu tiên)

h, $S = 2^2 + 4^2 + 6^2 + \ldots + n^2$ (Tính tổng các bình phương của n số tự nhiên chẵn đầu tiên)

i, $S = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

k, $S = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2}$

l, $S = \frac{1}{1.3} + \frac{1}{2.4} + \frac{1}{3.5} + \cdots + \frac{1}{n.(n+2)}$

Bài 6: Cho hai số nguyên dương m và n (m < n). Tính và đưa ra màn hình tổng các số chia hết cho 3 hoặc 5 trong phạm vi từ m đến n ($0 < m < n \leq 10^5$).

Bài 7: Tính giai thừa n! (n! = 1.2.3…. . n) của một số nguyên n

Bài 8: In ra bảng cửu chương của một số k bất kỳ.

Bài 9: In ra các ước nguyên dương của số nguyên dương n

Bài 10: Đếm số lượng các ước nguyên dương của số nguyên dương n

Bài 11: Tính tổng các ước nguyên dương của số nguyên dương n

Bài 12: Tìm ước chung của hai số nguyên dương m và n

Bài 13: Tìm ƯCLN và BCNN của hai số nguyên m và n

Bài 14: Kiểm tra số nguyên dương n có phải là số chính phương không.

Giải thích: Số chính phương là số tự nhiên có thể biểu diễn dưới dạng bình phương của một số nguyên ($n = k^2$)

Bài 15: Kiểm tra số nguyên n có phải là số hoàn hảo (hoàn thiện/hoàn chỉnh) không.

Giải thích: Số hoàn hảo (hay số hoàn thiện, số hoàn chỉnh) là số nguyên dương bằng tổng tất cả các ước số thực sự của nó (ước thực sự là các ước số dương của nó, ngoại trừ chính nó)

Bài 16: Kiểm tra số nguyên dương n có phải là số nguyên tố không (dùng for ... else).

Giải thích: Số nguyên tố là một số nguyên dương lớn hơn 1 chỉ có đúng hai ước là 1 và chính nó. Nói cách khác, một số n là số nguyên tố nếu n không thể chia hết cho bất kỳ số nào khác ngoài 1 và chính nó.

Bài 17: In ra dãy Fibonacci gồm n số hạng đầu tiên.

Giải thích: Dãy số Fibonacci là một dãy số tự nhiên bắt đầu bằng 0 và 1, trong đó mỗi số tiếp theo bằng tổng hai số trước đó, Công thức tổng quát của dãy Fibonacci được định nghĩa như sau:

$F_0 = 0$; $F_1 = 1$; $F_n = F_{n-1} + F_{n-2}$

Bài 18: Vẽ tam giác vuông bằng dấu * có chiều cao n.

```
*
**
***
****
```

Bài 19: Vẽ tam giác cân bằng dấu * có chiều cao n.

```
    *
   ***
  *****
*******
```

Lưu ý: Với mỗi biến m, n sử dụng trong các bài tập trên, ta làm theo hai trường hợp:

1, Với m, n nhập vào từ bàn phím (nhập/xuất dữ liệu chuẩn)

2, Với m, n đọc từ file input, kết quả xuất ra file output (đọc/ghi file)