

NHẬP MÔN LẬP TRÌNH (23CLC05)

Buổi 5 – 30/10/23

Vòng lặp lồng là gì?

Xuất “tam giác vuông kích thước n ”	
<pre>* ** *** **** ***** *****</pre>	
Mã giả tổng thể	Mã giả chi tiết
Nhập kích thước n Với dòng $i = 1, 2, \dots, n$: <ul style="list-style-type: none">• Xuất i dấu sao• Xuống dòng	Nhập kích thước n Với dòng $i = 1, 2, \dots, n$: <ul style="list-style-type: none">• Với $j = 1, 2, \dots, i$<ul style="list-style-type: none">○ Xuất dấu sao• Xuống dòng

Vòng lặp lồng là gì?

Lưu đồ	Python
<pre>graph TD Start([Nhập n]) --> I1[i = 1] I1 --> Cond1{i <= n?} Cond1 -- Đ --> J1[j = 1] Cond1 -- S --> Exit1[] J1 --> Cond2{j <= i?} Cond2 -- Đ --> PrintStar[/Xuất */] PrintStar --> JInc[j++] JInc --> Cond2 Cond2 -- S --> NewLine[/Xuống dòng/] NewLine --> IInc[Tăng i] IInc --> Cond1 Exit1 --> End([])</pre>	<pre>n = int(input("n = ? ")) for i in range(1, n + 1): for j in range(i): # Xuất i dấu sao print("*", end="") print()</pre> <p>C/C++</p> <pre>#include <iostream> using namespace std; int main() { int n; cout << "n = ? "; cin >> n; for (int i = 1; i <= n; i++) { // Xuất i dấu sao for (int j = 1; j <= i; j++) cout << "*"; cout << endl; } }</pre>

Vòng lặp lồng là gì?

- Khi thân của một lệnh lặp chứa một lệnh lặp khác, ta có cấu trúc **lặp lồng** (nested loop)
- Vòng lặp lồng thường mang lại cách xử lý mạnh mẽ
- Biến lặp của các vòng lặp lồng cần được đặt tên khác nhau (thường là i, j, k, ...)

Cộng điểm

- Làm bài tập 3.2.1 (Tài liệu C)

Thực thi vòng lặp lồng có nhanh không?

Tính tổng	
Cho x, n , tính $S = 1 + x + x^2 + \dots + x^n = \sum_{i=0}^n x^i$	
Mã giả tổng thể	Mã giả chi tiết
Nhập x, n $S = 0$ Với $i = 0, 1, \dots, n$: <ul style="list-style-type: none">• Tính $t = x^i$• $S = S + t$ Xuất S	Nhập x, n $S = 0$ Với $i = 0, 1, \dots, n$: <ul style="list-style-type: none">• $t = 1$• Với $j = 1, \dots, i$:<ul style="list-style-type: none">◦ $t = t \times x$• $S = S + t$• Xuất S

Thực thi vòng lặp lồng có nhanh không?

Python	C/C++
<pre>x = float(input("x = ? ")) n = int(input("n = ? ")) S = 0 for i in range(0, n + 1): term = 1 for j in range(i): term *= x S += term print(S)</pre>	<pre>#include <iostream> using namespace std; int main() { double x; int n; cout << "x = ? "; cin >> x; cout << "n = ? "; cin >> n; double S = 0; for (int i = 0; i <= n; i++) { double term = 1; for (int j = 1; j <= i; j++) term *= x; S += term; } cout << S; }</pre>

Khử lồng là gì?

Tính tổng	
Cho x, n , tính	
$S = 1 + x + x^2 + \dots + x^n = \sum_{i=0}^n x^i$	
Vòng lặp lồng	Vòng lặp đơn
Nhập x, n $S = 0$ Với $i = 0, 1, \dots, n$: <ul style="list-style-type: none">• $t = 1$• Với $j = 1, \dots, i$:<ul style="list-style-type: none">◦ $t = t \times x$• $S = S + t$ Xuất S	Nhập x, n $t = 1, S = t$ Với $i = 1, \dots, n$: <ul style="list-style-type: none">• $t = t \times x$• $S = S + t$ Xuất S
“Chiến lược” là tính số hạng sau của tổng dựa trên số hạng trước đó: số hạng thứ i (x^i) là tích của số hạng thứ $i - 1$ với x ($x^i = x^{i-1} \times x$). Ta cũng tách riêng số hạng đầu ($x^0 = 1$).	

Khử lồng là gì?

Mã giả	Python (đẹp)
Cho x, n $S = 1 + x + x^2 + \dots + x^n = \sum_{i=0}^n x^i$	<pre>x = float(input("x = ? ")) n = int(input("n = ? ")) S = sum([x**i for i in range(n + 1)]) print(S)</pre>
Python	C/C++
<pre>x = float(input("x = ? ")) n = int(input("n = ? ")) term, S = 1, 1 for i in range(1, n + 1): term *= x S += term print(S)</pre>	<pre>#include <iostream> using namespace std; int main(){ double x; int n; cout << "x = ? "; cin >> x; cout << "n = ? "; cin >> n; double term = 1, S = 1; for (int i = 1; i <= n; i++) { term *= x; S += term; } cout << S; }</pre>

Khử lồng là gì?

- Vòng lặp lồng “đôi khi” kém hiệu quả!
- Việc chuyển các vòng lặp lồng về vòng lặp đơn (tức không lồng) được gọi là **“khử lồng”**
- Xử lý không lồng thường hiệu quả nhưng phức tạp hơn so với xử lý lồng

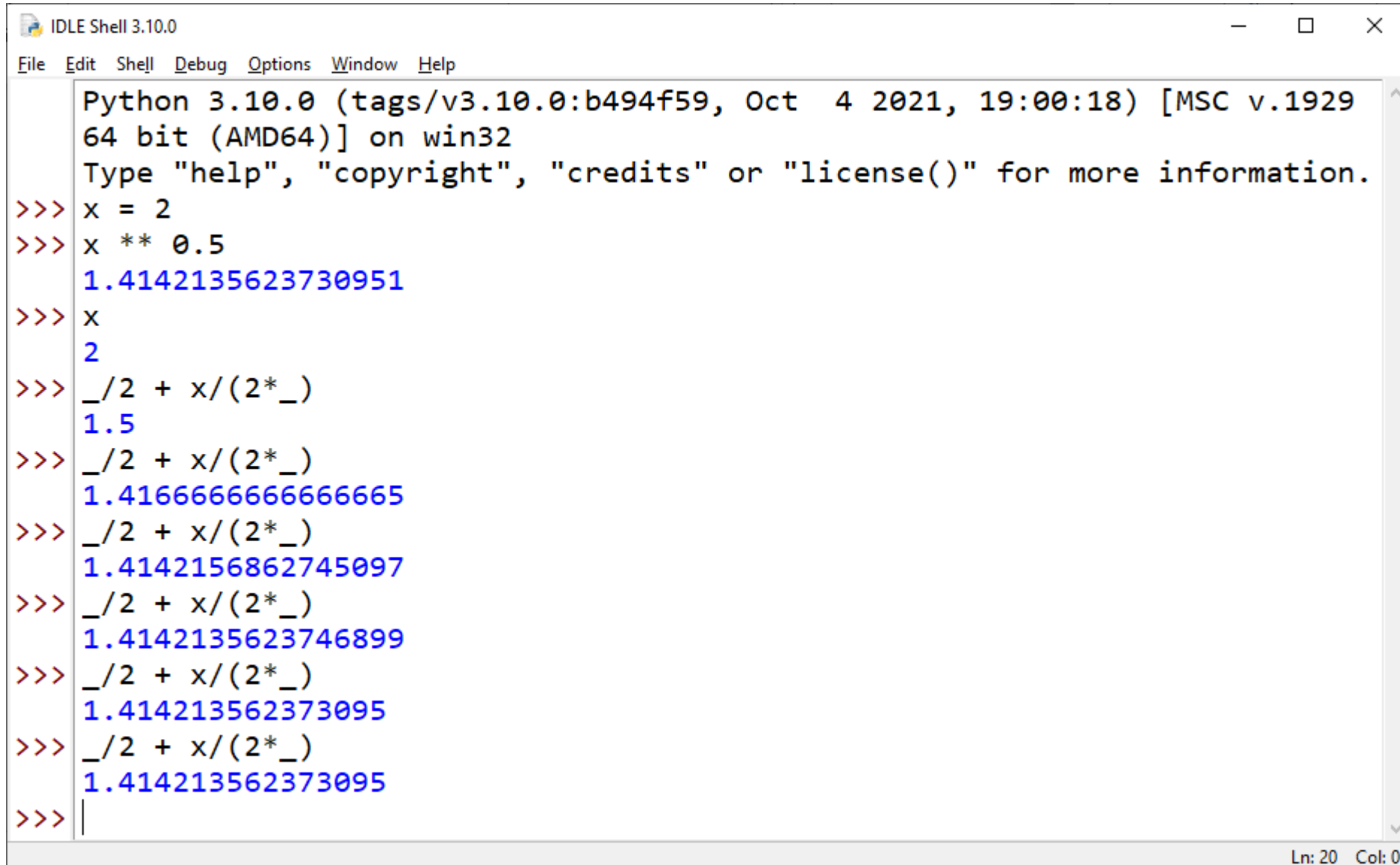
Cộng điểm

- Viết chương trình cho bài tập 2.5.1b, 2.5.2b (Tài liệu C) không dùng vòng lặp lồng

Giải lao!!!

- Giải lao đến 9h50

Làm sao viết hàm tính căn?



```
IDLE Shell 3.10.0
File Edit Shell Debug Options Window Help
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> x = 2
>>> x ** 0.5
1.4142135623730951
>>> x
2
>>> _/2 + x/(2*_ )
1.5
>>> _/2 + x/(2*_ )
1.4166666666666665
>>> _/2 + x/(2*_ )
1.4142156862745097
>>> _/2 + x/(2*_ )
1.4142135623746899
>>> _/2 + x/(2*_ )
1.414213562373095
>>> _/2 + x/(2*_ )
1.414213562373095
>>> |
```

Ln: 20 Col: 0

Làm sao viết hàm tính căn?

Python	C/C++
<pre>import math def mysqrt(x): y = x for i in range(100): y = y/2 + x/(2*y) return y print(mysqrt(2)) print(mysqrt(4)) x = float(input("Enter x > 0: ")) print(mysqrt(x)) print(math.sqrt(x))</pre>	<pre>#include <iostream> #include <cmath> using namespace std; double mysqrt(double x) { double y = x; for (int i = 1; i <= 100; i++) y = y/2 + x/(2*y); return y; } int main() { cout << mysqrt(2) << endl; cout << mysqrt(4) << endl; double x; cout << "Enter x > 0: "; cin >> x; cout << mysqrt(x) << endl; cout << sqrt(x) << endl; }</pre>

Hàm là gì?

- **Hàm** (function) là một khối lệnh được đặt tên và tham số hóa
- Hàm giúp không phải “chép-sửa” một khối lệnh nhiều chỗ
 - 👉 định nghĩa một lần (một chỗ) và dùng nhiều lần (nhiều chỗ)!
- Hàm là công cụ quan trọng nhất của lập trình
- Hầu hết các ngôn ngữ lập trình đều cung cấp rất nhiều hàm được định nghĩa sẵn trong các module, thư viện

Định nghĩa hàm là gì?

- Hàm được mô tả qua **định nghĩa hàm** (function definition) gồm **tên hàm** (function name), các **tham số** (parameter) và **thân hàm** (function body), chính là khối lệnh của hàm
- C/C++ yêu cầu phải cho biết kiểu của các tham số và kiểu của **giá trị trả về** (return value) của hàm
- Khai báo gồm tên hàm và các tham số (cùng với kiểu các tham số và kiểu giá trị trả về) được gọi là **nguyên mẫu** (prototype) của hàm
- Các tham số còn được gọi là **đầu vào** (input) và giá trị trả về còn được gọi là **đầu ra** (output) của hàm

Lời gọi hàm là gì?

- Hàm được dùng (tức là gọi chạy hay thực thi thân hàm) qua **lời gọi hàm** (function call) gồm tên hàm xác định hàm được gọi và các **đối số** (argument) xác định các giá trị truyền (gán) cho các tham số tương ứng trước khi thực thi thân hàm
- Lời gọi hàm là một biểu thức mà giá trị chính là giá trị trả về của hàm sau khi hàm thực thi xong (nếu có)
- Các hàm phải được định nghĩa trước khi dùng

Tham số khác đối số thế nào?

- Tham số (parameter) xuất hiện trong định nghĩa hàm như là biến chứa giá trị đầu vào của hàm khi hàm được gọi thực thi
- Đối số (argument) xuất hiện trong lời gọi hàm như là biểu thức mà giá trị sẽ được gán cho tham số trước khi thân hàm được thực thi
- Để dễ phân biệt, tham số còn được gọi là tham số/đối số **hình thức** (formal parameter/argument) và đối số còn được gọi là tham số/đối số **thực tế** (actual parameter/argument)

Cộng điểm

- Làm bài tập 3.3.1 (Tài liệu C)

Hàm có nhất thiết trả về giá trị hay không?

Python	C/C++
<pre>def hello(name): hello_string = "Hi " + name length = len(hello_string) print("=" * (length + 4)) print(" " + hello_string + " ") print("=" * (length + 4))</pre>	<pre>#include <iostream> using namespace std; void hello(string name) { string hello_string = "Hi " + name; int length = hello_string.length(); for (int i = 1; i <= length + 4; i++) cout << "="; cout << endl; cout << " " << hello_string << " \n"; for (int i = 1; i <= length + 4; i++) cout << "="; cout << endl; } int main() { hello("Python"); hello("Guido van Rossum"); string a_name; cout << "What's your name? "; getline(cin, a_name); hello(a_name); }</pre>

Thủ tục là gì?

- Các hàm có trả về giá trị thường được gọi là hàm tính toán còn các hàm không trả về giá trị thường được gọi là **thủ tục** (procedure)
- Trong C/C++, từ khóa **void** được dùng để mô tả rằng hàm không trả về giá trị; khi đó lời gọi hàm tương ứng là một **biểu thức không giá trị** (void expression)
- Trong Python, mọi hàm đều trả về giá trị và giá trị trả về của thủ tục là **None** (giá trị duy nhất của kiểu **NoneType**)
- Thủ tục thường mô tả “làm gì đó” (chứ không phải “tính gì đó” như các hàm trả về giá trị)

Lệnh return là gì?

- **Lệnh return** (return statement) yêu cầu hàm kết thúc thực thi và trả về giá trị tương ứng nếu có
- Hàm cũng có thể kết thúc “tự nhiên” khi thực thi xong thân hàm (là khối lệnh)
- Hàm tính toán phải dùng “lệnh return có đối số” để trả về giá trị cho hàm
- Thủ tục có thể dùng “lệnh return không đối số” để kết thúc “sớm”

Cộng điểm

- Làm bài tập 3.4.1 (Tài liệu C) và mở rộng phù hợp