**NGUYỄN TẤN PHONG**

**TÀI LIỆU**

HƯỚNG DẪN HỌC TIN HỌC

**2**

**BÀI 1**

**MÁY TÍNH**

**VÀ CHƯƠNG TRÌNH MÁY TÍNH**

***- Con người chỉ dẫn cho máy tính thực hiện công việc thông qua các lệnh***

***- Chương trình là bản hướng dẫn cho máy tính thực hiện những nhiệm vụ cụ thể***

**1. Viết chương trình - ra lệnh cho máy tính làm việc**

Để máy tính thực hiện một công việc nào đó, con người cần đưa cho máy tính các chỉ dẫn thích hợp (câu lệnh). Máy tính sẽ lần lượt thực hiện các lệnh đó giống như rô-bốt ở ví dụ trên.

Để tránh “nhắc” rô-bốt thực hiện từng câu lệnh, người ta thường tập hợp các lệnh đó và lưu trong rô-bốt với tên “Hãy gỡ bom”. Khi đó chỉ cần ra lệnh “Hãy gỡ bom” thì rô-bốt sẽ tự động thực hiện các lệnh đó. Việc tập hợp các lệnh để điều khiển rô-bốt như vậy chính là viết chương trình. Tương tự, để điều khiển máy tính làm việc, chúng ta cũng viết chương trình máy tính.

***Chương tình máy tính là một dãy các câu lệnh mà máy tính có thể hiểu và thực hiện được.***

***Tại sao cần viết chương trình?***

Trong thực tế các công việc con người muốn máy tính thực hiện rất đa dạng và phức tạp. Một lệnh đơn giản không đủ để chỉ dẫn cho máy tính hoàn thành công việc. Vì thế, để khai thác triệt để tốc độ của máy tính việc viết nhiều lệnh và tập hợp lại trong một chương trình giúp con người điều khiển máy tính một cách đơn giản và hiệu quả hơn.

Khi thực hiện chương trình, máy tính sẽ thực hiện các câu lệnh trong chương tình một cách tuần tự, nghĩa là thực hiện xong một lệnh sẽ thực hiện lệnh tiếp theo, từ lệnh đầu tiên đến lệnh cuối cùng.

**3**

Trở lại ví dụ về rô-bốt gỡ bom, chương trình có thể có các lệnh sau:

Hãy gỡ bom Bắt đầu

Quay phải Tiến 1 bước. Quay trái Tiến 3 bước. Gỡ bom.

Kết thúc

**2. Chương trình và ngôn ngữ lập trình**

Chúng ta đã biết, để máy tính có thể xử lí, thông tin đưa vào máy phải được chuyển đổi thành dạng dãy bit (dãy các số chỉ gồm 0 và 1). Các dãy bit là cơ sở để tạo ra ngôn ngữ dành cho máy tính, được gọi là *ngôn ngữ máy*. Những chương trình máy tính đầu tiên khi máy tính mới xuất hiện được viết chính bằng ngôn ngữ này.

Tuy nhiên, việc viết chương trình bằng ngôn ngữ máy rất khó khăn và mất nhiều thời gian, công sức. Bởi lẽ, về mặt trực quan, các câu lệnh được viết dưới dạng dãy bit khác xa ngôn ngữ tự nhiên nên khó nhớ, khó sử dụng. Vì vậy, người ta mong muốn có thể sử dụng được các từ có nghĩa, dễ hiểu và dễ nhớ để viết các câu lệnh thay cho các dãy bit khô khan. Các *ngôn ngữ lập trình* đã ra đời để phục vụ mục đích đó.

***Ngôn ngữ lập tình là ngôn ngữ dùng để viết các chương tình máy tính.***

Như vậy, để tạo ra chương trình máy tính, chúng ta phải viết chương trình bằng một ngôn ngữ nào đó. Nói cách khác, ngôn ngữ lập trình là công cụ giúp để tạo ra các chương tình máy tính.

Tuy nhiên, máy tính vẫn chưa thể hiểu được các chương trình được viết bằng ngôn ngữlậptrình. Chươngtìnhcòncầnđược chuyểnđổisangngônngữ máybằngmột *chương trình dịch* tương ứng.

**4**

Như vậy, việc tạo ra chương tình máy tính thực chất gồm 2 bước sau: *Viết* chương trình bằng một ngôn ngữ lập trình.

*Dịch* chương tình thành ngôn ngữ máy để máy tính có thể hiểu được.

***Chương trình viết bằng ngôn ngữ lập trình cần được chuyển đổi thành ngôn ngữ máy***

**Viết chương trình**

**Dịch**

Kết quả nhận được sau bước là danh sách các lệnh được lưu thành một tệp văn bản trong máy tính; còn kết quả của bước là một tệp có thể thực hiện trên máy tính. Các tệp đó thường được gọi chung là chương trình.

Chương trình có thể được soạn thảo nhờ một chương trình soạn thảo (tương tự như phần mềm soạn thảo Word). Chương trình soạn thảo và chương trình dịch cùng với các công cụ trợ giúp tìm kiếm, sửa lỗi và thực hiện chương trình thường được kết hợp vào một phần mềm, được gọi là *môi trường lập trình*. Ví dụ, với ngôn ngữ lập trình **Python** có môi trường lập trình phổ biến là **IDLE (Python 3.8 32-bit)**.

CÂU HỎI VÀ BÀI TẬP

1. Hãy cho biết chương trình máy tính là gì?

2. Hãy cho biết ngôn ngữ lập trình là gì? Ngôn ngữ máy là gì? 3. Hãy cho biết chương trình dịch là gì?

**5**

**BÀI 2**

**LÀM QUEN VỚI CHƯƠNG TRÌNH VÀ NGÔN NGỮ LẬP TRÌNH**

***- Ngôn ngữ lập trình là gì?***

***- Từ khóa của ngôn ngữ lập trình***

***- Cấu trúc của một chương trình máy tính.***

**1. Ngôn ngữ lập trình gồm những gì?**

Chúng ta đã biết chương trình có thể có nhiều câu lệnh. Các câu lệnh được viết từ những kí tự nhất định. Tập kí tự này tạo thành ***bảng chữ cái*** của ngôn ngữ lập trình.

Giống như ngôn ngữ tự nhiên, mọi ngôn ngữ lập trình đều có bảng chữ cái riêng. Các câu lệnh chỉ được viết từ các chữ cái của bảng chữ cái đó.

Bảng chữ cái của các ngôn ngữ lập trình thường gồm các chữ cái tiếng Anh và một số kí hiệu khác như dấu phép toán (+, -, \*, /,...), dấu đóng mở ngoặc, dấu nháy,... Nói chung, hầu hết các kí tự có trên bàn phím máy tính đều có mặt trong bảng chữ cái của mọi ngôn ngữ lập trình.

Câu lệnh trong chương trình trên gồm các từ và các kí hiệu được viết theo một quy tắc nhất định. ***Các quy tắc*** này quy định cách viết các từ và thứ tự của chúng. Chẳng hạn, trong ví dụ trên các từ được cách nhau bởi một hoặc nhiều dấu cách, câu lệnh **for** được kết thúc bởi dấu hai chấm (:), dòng lệnh print('Chào các bạn lớp ',i) được viết thụt vào trong lệnh **for** và có cụm từ tiếng Việt trong cặp dấu nháy đơn (' '),... Nếu câu lệnh bị viết sai quy tắc, chương trình dịch sẽ nhận biết và thông báo lỗi.

Mặt khác, mỗi câu lệnh đều có một ý nghĩa riêng xác định các thao tác mà máy tính cần thực hiện. Dòng lệnh đầu tiên trong ví dụ trên là lệnh khai báo một danh sách các

**6**

lớp học của khối lớp 8. Câu lệnh **for *i* in *danhsáchlớp:*** là lệnh lặp với số lần tương ứng với các lớp trong sách lớp đã khai báo, ….

Tóm lại, về cơ bản ngôn ngữ lập trình gồm ***bảng chữ cải*** và ***các quy tắc*** để viết các câu lệnh có ý nghĩa xác định, cách bố trí các câu lệnh,... sao cho có thể tạo thành một chương trình hoàn chỉnh và thực hiện được trên máy tính.

**2. Từ khóa và tên**

Trong chương trình trên, ta thấy có ba từ **for, in, print** đó là những ***từ khoá*** được quy định tuỳ theo mỗi ngồn ngữ lập trình. Từ khoá của một ngôn ngữ lập trình là những *từ dành riêng,* không được dùng các từ khoá này cho bất kì mục đích nào khác ngoài mục đích sử dụng do ngôn ngữ lập trình quy định.

Ngoài các từ khoá, chương trình trong ví dụ trên còn có từ như ***danhsáchlớp*** đó là các ***tên*** được dùng trong chương trình. Khi viết chương trình để giải các bài toán, ta thường thực hiện tính toán với những đại lượng (ví dụ như so sánh chiều cao, tính điểm trung bình,...) hoặc xử lí các đối tượng khác nhau. Các đại lượng và đối tượng này đều phải được đặt tên. Ví dụ tên ***danhsáchlớp*** là do người lập trình đặt để chỉ một tập hợp có các phần tử là 8A1, 8A2, 8A3.

Tên do người lập trình đặt phải tuân thủ các quy tắc của ngôn ngữ lập trình cũng như của chương trình dịch và thoả mãn:

*- Tên khác nhau* tương ứng với những *đại lượng khác nhau.* - Tên *không được trùng với các từ khoá.*

Tên trong chương trình được dùng để phân biệt và nhận biết các đại lượng khác nhau. Do vậy, tuy có thể đặt tên tuỳ ý, nhưng để dễ sử dụng nên đặt tên sao cho ***ngắn gọn, dể nhớ*** và ***dể hiểu***.

***Ví dụ:*** Tên hợp lệ trong ngôn ngữ lập trình Python không được bắt đầu bằng chữ số và không được chứa dấu cách (kí tự trống). Do vậy chúng ta có thể đặt tên *STamgiac* để chỉ diện tích hình tam giác, hoặc đặt tên *ban\_kinh* cho bán kính của hình tròn,.... Các tên đó là những ***tên hợp lệ***, còn các tên *Lop em, 8A,...* là những tên không hợp lệ. Nếu để trong cặp dấu nháy đơn như **'8A1', '8A2', '8A3'** thì là các chuỗi.

Chúng ta sẽ dần làm quen với cách đặt tên và sử dụng tên trong các bài sau. **3. Ví dụ về ngôn ngữ lập trình**

Trong phần này chúng ta sẽ làm quen với một ngôn ngữ lập trình cụ thể, ngôn ngữ Python. Để lập trình bằng ngôn ngữ Python, máy tính cần được cài đặt môi trường lập trình trên ngôn ngữ này.

Dưới đây là minh hoạ việc viết và chạy một chương trình cụ thể trong môi trường lập trình ***Python 3.8.5.***

Khi khởi động phần mềm ***IDLE (Python 3.8 32-bit)***, cửa sổ soạn thảo chương trình như hình 8 dưới đây:

**7**

Ta sử dụng bàn phím để soạn thảo chương trình tương tự như soạn thảo văn bản với Word. Sau khi đã soạn thảo xong một câu lệnh (hoặc nhóm câu lệnh), nhấn tổ hợp phím Enter để dịch chương trình. Chương trình dịch sẽ kiểm tra các lỗi chính tả và cú pháp; nếu gặp câu lệnh sai, chương trình dịch sẽ thông báo để người viết chương trình dễ nhận biết và chỉnh sửa. Nếu đã hết lỗi, sau khi dịch, màn hình có dạng như hình 9 dưới đây:

Để có thể soạn thảo chương trình và lưu lại như thành một tập tin ta thực hiện soạn thảo trong môi trường soạn thảo như sau:

**8**

Sau khi soạn thảo xong ta nhấn phím F5 để dịch và chạy chương trình như hình dưới:

CÂU HỎI VÀ BÀI TẬP

1. Hãy cho biết các thành phần cơ bản của một ngôn ngữ lập trình?

2. Cho biết sự khác nhau giữa tù khoá và tên. Cho biết cách đặt tên trong chương trình.

3. Trong các tên sau đây, tên nào là hợp lệ trong ngôn ngữ Python?

A) a;

E) beginprogram;

B) Tamgiac; C) 8a;

F) end; G) b1;

D) Tam giac;

H) abc

**9**

**BÀI THỰC HÀNH**

**1**

**LÀM QUEN VỚI PYTHON**

***- Bước đầu làm quen với môi trường Python, nhận diện màn hình soạn thảo, cách mở các bảng chọn và chọn lệnh.***

***- Gõ được một chương trình Python đơn giản.***

***- Biết cách dịch, sửa lỗi trong chương trình, chạy chương trình và xem kết quả.***

**NỘI DUNG *Bài 1:* Hướng dẫn cài đặt Python 3.x.x Shell**

**a/ Tải phần mềm Python 3.x.x Shell**

Địa chỉ: <https://www.python.org/downloads/>

***Chú ý quan trọng:*** Python hiện nay có hai phiên bản là 2.x và 3.x; phiên bản 2.x đã dừng hỗ trợ vào 01.01.2020; phiên bản 3.x thì vẫn tiếp tục được phát triển với nhiều thư viện. Việc chọn lựa phiên bản 2.x hay 3.x cũng có những sự khác biệt về cách trình bày câu lệnh và khả năng hỗ trợ lập trình đa nền tảng khác nhau.

**b/ Cài đặt Python 3.8.5**

Khi tải được tập tin có chữ **“python-3.8.5.exe”** về, hãy nhấp đôi chuột vào tập tin để cài đặt. Việc cài đặt rất đơn giản và nhanh chóng. ***Hãy để tất cả các option mặc định và chạy chương trình.***

Khi cài đặt xong trên màn hình có icon như hình:

+ IDLE (Python 3.8 32-bit) là môi trường lập trình kết hợp viết code thành các tập tin \*.py và viết lệnh thực thi ngay (gọi là ***coding mode – giao diện soạn thảo lệnh;*** *trong chương trình tin học 8 chúng ta chỉ thực hành với môi trường IDLE này*).

**10**

+ Python 3.8 (32-bit) là môi trường lập trình viết lệnh thực thi ngay (gọi là ***interactive mode – giao diện tương tác***).

**c/ Giao diện của Python Python 3.x.x Shell *c1. Giao diện tương tác (interactive mode)***

Trong cửa sổ bên trên chúng ta thấy có 2 phần chính:

***(1)Thanh công cụ* (menu)**: nó bao gồm rất nhiều, nhưng chỉ một số chúng sẽ được sử dụng thường xuyên.

***(2) Phần trung tâm soạn thảo lệnh***: đây là nơi bạn viết code python ở chế độ chạy lệnh trực tiếp (**interactive mode)** và xem kết quả thực thi chương trình (bao gồm kết quả tính toán và thông báo lỗi)

***c2. Giao diện soạn thảo lệnh (coding mode)***

Bây giờ từ giao diện interactive mode, hãy chọn File => New (hoặc Ctrl + N)

Hãy gõ những dòng lệnh như hình. Gõ xong, nhấn Ctrl + S (hoặc File => Save). Một hộp thoại xuất hiện như hình:

**11**

Gõ vào tên chương trình là ***ctdautien.py*** (***.py*** là đuôi mở rộng của tập tin Python) Hãy nhấn phím **F5** và xem kết quả ở cửa sổ ***interactive mode***.

***Hãy đặc biệt ghi nhớ là nhấn phím* F5 *để dịch và chạy chương trình.* c.3/ Tìm hiểu “chương trình đầu tiên” có gì?**

Hãy quan sát nội dung đoạn code mà bạn đã tạo như hình: **Câu lệnh trong python** **Giải thích**

Chú thích trong chương trình, bắt đầu bằng dấu ***#***

Khai báo biến ten, chờ nhận dữ liệu từ bàn phím

In ra dòng chữ: Xin chào bạn ….

***Bây giờ hãy nhấn* Ctrl + Q *để thoát Python.***

***Bài 2:* Soạn thảo, lưu, dịch và chạy một chương trình đơn giản:**

a. Khởi động lại Python, vào giao diện ***coding mode*** và gõ các dòng lệnh dưới đây:

b. Nhấn phím F5 (hoặc File => Save) để lưu chương trình. Khi hộp thoại hiện ra hãy gõ tên tập tin là ***chaolop.py*.**

c. Quan sát kết quả ở màn hình ***interactive mode***

***Lưu ý:*** Có thể dùng bảng chọn Run để chạy chương trình.

**12**

***Bài 3:* Tìm hiểu một số lỗi trong chương trình và thông báo lỗi:**

a. Hãy sửa tên ***danhsachlop*** thành ***danh sách lớp***. Chạy chương trình và quan sát thông báo lỗi như hình :

***Giải thích lỗi:*** tên ***danh sách lớp*** do người dùng đặt có chứa khoảng trắng.

Hãy xóa bỏ khoảng trắng, sửa lại thành ***danhsáchlớp***. Chạy chương trình và quan sát kết quả.

Nếu chỉ sửa dòng lệnh đầu tiên là ***danhsáchlớp*** và dòng lệnh thứ hai vẫn là ***danhsachlop*** như hình thì kết quả sẽ báo lỗi như sau :

***Giải thích lỗi:*** ở cửa sổ ***interactive mode*** xuất hiện thông báo lỗi màu đỏ, tạm dịch là: “tại dòng 2 trong mô đun, dòng lệnh **for i in danhsachlop:**, tên ‘danhsachlop’ chưa định nghĩa”.

b. Xóa bỏ một từ khóa trong đoạn chương trình trên. Chạy chương trình và quan sát lỗi.

**13**

***Bài 4:* Chỉnh sửa chương trình:**

a. Hãy chỉnh sửa để in ra lời chào và tên của 5 bạn trong tổ của em, ví dụ :

b. Nhấn Ctrl + Shift + S (hoặc File => Save As…), gõ tên tập tin ***chaoban.py***

**TỔNG KẾT**

1. Các bước đã thực hiện: Khởi động Python Soạn thảo chương trình

Biên dịch và chạy chương trình: F5 Lưu chương trình

2. Các từ khóa của Python trong bài học là: **for**, **in**, **print**

3. Dấu phẩy (,) dùng để phân cách các giá trị trong lệnh **print** 4. Dấy nháy đơn ('') để chỉ chuỗi.

5. Lệnh **print()** để in thông báo ra màn hình và xuống dòng.

**14**

**BÀI 3**

**CHƯƠNG TRÌNH MÁY TÍNH VÀ DỮ LIỆU**

***Một số kiểu dữ liệu cơ bản trong ngôn ngữ lập trình.***

***Tương tác người-máy.***

**1. Dữ liệu và kiểu dữ liệu**

Máy tính là công cụ xử lí thông tin, còn chương trình chỉ dẫn cho máy tính cách thức xử lí thông tin để có kết quả mong muốn. Thông tin rất đa dạng nên dữ liệu trong máy tính cũng rất khác nhau về bản chất. Để dễ dàng quản lí và tăng hiệu quả xử lí, các ngôn ngữ lập trình thường phân chia dữ liệu thành các *kiểu* khác nhau: chữ, số nguyên, số thập phân,...

***Ví dụ 1.*** Hình dưới đây minh hoạ kết quả thực hiện của một chương trình: in ra màn hình với các kiểu dữ liệu quen thuộc là chữ và số.

Các kiểu dữ liệu thường được xử lí theo các cách khác nhau. Chẳng hạn, ta có thể thực hiện các phép toán số học với các số, nhưng với các kí tự hay xâu kí tự thì các phép toán đó không có nghĩa.

Các ngôn ngữ lập trình định nghĩa sẵn một số kiểu dữ liệu cơ bản. Kiểu dữ liệu xác định miền giá trị có thể của dữ liệu và các phép toán có thể thực hiện trên các dữ liệu đó. Dưới đây là một số kiểu dữ liệu thường dùng nhất:

***Số nguyên****,* ví dụ số học sinh của một lớp, số sách trong thư viện,... ***Số thực****,* ví dụ chiều cao của bạn Bình, điểm trung bình môn Toán,...

***Kí tự*** là một chữ, chữ số hay kí hiệu đặc biệt khác, ví dụ: 'a', 'A', '+', '1' (chữ số 1, khác với số nguyên 1), ' ' (kí tự trống),... Trong đa số các trường hợp, kí tự thường là một *'chữ cái'* của ngồn ngữ lập trình.

**15**

***Xâu kí tự*** (hay *xâu* hoặc *chuỗi)* là dãy các *'chữ cái'* lấy từ bảng chữ cái của ngôn ngữ lập trình, ví dụ: *"Chao cac ban", "Lop 8E", "*2/9/1 945*"…*

Trong các ngôn ngữ lập trình, dữ liệu kiểu số nguyên còn có thể được phân chia tiếp thành các kiểu nhỏ hơn theo các phạm vi giá trị khác nhau, dữ liệu kiểu số thực còn có thể được phân chia thành các kiểu có độ chính xác (số chữ số thập phân) khác nhau.

Ngoài các kiểu nói trên, mỗi ngôn ngữ lập trình cụ thể còn định nghĩa nhiều kiểu dữ liệu khác. Số các kiểu dữ liệu và tên kiểu dữ liệu trong mỗi ngôn ngữ lập trình có thể khác nhau.

***Ví dụ 2.*** Bảng dưới đây liệt kê một số kiểu dữ liệu cơ bản của ngôn ngữ lập trình Python:

**Tên kiểu** **Kí hiệu** **Ví dụ** **Phạm vi giá trị**

**Số nguyên int Số thực** **float Xâu** **str**

12; -15; 17878; … Không bị giới hạn, phụ thuộc

1.2; 9.0; -2.3; … vào tài nguyên của máy tính. "Lập trình Python”

Trong Python, để chỉ rõ cho chương trình dịch hiểu dãy *'chữ cái'* là kiểu xâu, ta phải đặt dãy *'chữ cái'* đó trong cặp dấu nháy đơn hoặc nháy kép. Ví dụ: 'A5324', '863', "A5324", "8635465465".

**2. Các phép toán với kiểu dữ liệu số**

Trong mọi ngôn ngữ lập trình ta đều có thể thực hiện các phép toán số học cộng, trừ, nhân và chia với các số nguyên và số thực.

Chẳng hạn, bảng dưới đây là kí hiệu của các phép toán số học đó trong ngôn ngữ Python:

**Kí hiệu Phép toán Kiểu dữ liệu**

+ cộng

- trừ

\* nhân

/ chia

\*\* lũy thừa

// chia lấy phần nguyên

**int, float int, float int, float int, float int, float int**

% chia lấy phần dư **int**

Chúng ta đã quen thuộc với các phép toán cộng, trừ, nhân và chia. Tuy nhiên, hãy lưu ý rằng hầu hết các ngôn ngữ lập trình đều xem kết quả chia hai số *n* và *m* (tức *n/m*) là số thực, cho dù *n* và *m* là các số nguyên và *n* có thể chia hết cho *m*.

Sử dụng dấu ngoặc, ta có thể kết hợp các phép tính số học nói trên để có các biểu thức số học phức tạp hơn. Sau đây là một số ví dụ về biểu thức số học và cách viết chúng trong ngôn ngữ lập trình Python:

**16**

**Biểu thức số học Cách viết trong Python**

*ab**c**d* a\*b-c+d

155 2

*a*3  *b*5(*x*2)2

15+5\*(a/2)

(x+5)/(a+3)-y/(b+5)\*(x+2)\*\*2

Chú ý, trong Python khi viết các biểu thức toán phức tạp ta chỉ có thể dùng cặp dấu ngoặc tròn ( và ) để gộp các phép toán, không dùng các cặp dấu ngoặc {}, [].

**3. Các phép so sánh**

Ngoài các phép toán số học, ta còn thường ***so sánh*** các số.

Khi viết chương trình, để so sánh dữ liệu (số, biểu thức,...) chúng ta sử dụng các kí hiệu do ngôn ngữ lập trình quy định.

Kí hiệu các phép toán và phép so sánh có thể khác nhau, tuỳ theo từng ngôn ngữ lập trình.

Bảng dưới đây cho biết kí hiệu của các phép so sánh trong ngôn ngữ Python:

Phép so sánh

Bằng Khác Nhỏ hơn

Nhỏ hơn hoặc bằng Lớn hơn

Lớn hơn hoặc bằng

Kí hiệu toán học

= ≠ < ≤ >

≥

Kí hiệu trong Python

== != < <= >

>=

Ví dụ trong Python

5 == 5 6 != 5 3 < 5 5 <= 6 9 > 6

9 >= 6

Kết quả của phép so sánh chỉ có thể là ***đúng*** hoặc ***sai***. Ví dụ, phép so sánh 9 > 6 cho kết quả đúng, 10 == 9 cho kết quả sai hoặc 5 < 3 cũng cho kết quả sai,...

Để so sánh giá trị của hai biểu thức, chúng ta cũng sử dụng một trong các kí hiệu toán học trong bảng trên. Ví dụ:

So sánh biểu thức

5\*2 == 9

15 + 7 > 20 - 3

5 + x < 10

Kết quả phép so sánh

Sai Đúng

Đúng hoặc sai phụ thuộc vào giá trị cụ thể của x

**4. Giao tiếp người – máy tính**

Trong khi thực hiện chương trình máy tính, con người thường có nhu cầu can thiệp vào quá trinh tính toán, thực hiện việc kiểm tra, điều chỉnh, bổ sung. Ngược lại, máy tính cũng cho thông tin về kết quả tính toán, thông báo, gợi ý,... Quá trình trao đổi dữ liệu hai chiều như thế thường được gọi là giao tiếp hay tương tác giữa người và máy tính. Với các

**17**

máy tính cá nhân, tương tác người-máy thường được thực hiện nhờ các thiết bị chuột, bàn phím và màn hình. Dưới đây là một số trường hợp tương tác người-máy.

**a) Thông báo kết quả tính toán**

Thông báo kết quả tính toán là yêu cầu đầu tiên đối với mọi chương trình. Ví dụ, câu lệnh thông báo ra màn hình trong Python:

in kết quả tính phép toán 2007 + 5123 như hình dưới đây:

**b) Nhập dữ liệu**

Một trong những tương tác thường gặp là chương trình yêu cầu nhập dữ liệu. Chương trình sẽ tạm ngừng để chờ người dùng “***nhập dữ liệu***” từ bàn phím hay bằng chuột. Hoạt động tiếp theo của chương trình sẽ tuỳ thuộc vào dữ liệu được nhập vào.

Ví dụ, chương trình yêu cầu nhập năm sinh từ bàn phím. Khi đó ta cần gõ một số tự nhiên ứng với năm sinh. Câu lệnh thông báo ra màn hình và chờ nhập năm sinh vào biến ***ns*** trong Python:

Sau khi nhập một số, nhấn phím **Enter** để xác nhận, chương trình sẽ tiếp tục hoạt động.

**c) Hộp thoại**

Hộp thoại được sử dụng như một công cụ cho việc giao tiếp người-máy tính trong khi chạychương trình. Ví dụ, trong Python khi viết xong hoặc chỉnh sửa người dùng nhấn F5 để chạy chương trình thì một hộp thoại dạng sau đây có thể xuất hiện:

Khi đó, nếu nháy chuột vào nút OK, chương trình sẽ được lưu lại và dịch, nếu chọn Cancel sẽ huỷ lệnh trở về màn hình soạn thảo như bình thường.

CÂU HỎI VÀ BÀI TẬP

1.Hãy nêu ít nhất hai kiểu dữ liệu và một phép toán có thể thực hiện được trên một kiểu dữ liệu, nhưng phép toán đó không có nghĩa trên kiểu dữ liệu kia.

2. Dãy chữ số 2020 có thể thuộc những kiểu dữ liệu nào? 3. Hãy phân biệt ý nghĩa của các câu lệnh Python sau đây:

và

**18**

**BÀI THỰC HÀNH**

**2**

**VIẾT CHƯƠNG TRÌNH ĐỂ TÍNH TOÁN**

***- Luyện tập soạn thảo, chỉnh sửa chương trình, biên dịch, chạy và xem kết quả hoạt động của chương trình trong môi trường soạn thảo lệnh.***

***- Thực hành với các biểu thức số học trong chương trình Python.***

**NỘI DUNG**

***Bài 1:* Luyện tập và gõ các biểu thức số học trong chương trình Python:**

a) Viết các biểu thức toán học sau đây dưới dạng biểu thức trong Python :

A) 15.43012

105 18 (10 2)2 31 51 (31)

(102)2 24

(31)

b) Khởi động Python và gõ chương trình sau để tính các biểu thức trên :

c) Lưu chương trình với tên ***Bai1TH2.py***, dịch và chạy chương trình. Hãy kiểm tra kết quả nhận được trên màn hình.

***Bài 2:* Tìm hiểu các phép chia lấy phần nguyên và phép chia lấy phần dư với số nguyên trong chương trình Python :**

a) Mở tệp mới và gõ chương trình sau đây :

b) Lưu chương trình với tên ***Bai2TH2.py***, dịch và chạy chương trình. Quan sát các kết quả nhận được và cho nhận xét về các kết quả đó.

c) Hãy thêm lệnh **print()** vào sau mỗi câu lệnh trong chương trình trên. Dịch và chạy chương trình. Quan sát kết quả hoạt động của chương trình và cho nhận xét.

**19**

***Bài 3:* Tìm hiểu thêm về cách ghi dữ liệu ra màn hình trong chương trình Python:**

Mở lại tập tin ***Bai1TH2.py*** và lưu lại bản sao với tên ***Bai3TH2.py***. Sửa lại ba lệnh cuối thành :

Dịch và chạy chương trình. Quan sát kết quả trên màn hình và rút ra nhận xét của em.

**TỔNG KẾT**

1. Kí hiệu của các phép toán số học trong Python: +, -, \*, /, //, % và \*\*. 2. Lệnh **print()** để xuống một dòng trống.

3. Lệnh **round(*m*,*n*)** để làm tròn số thực ***m*** với ***n*** chữ số thập phân sau dấu phẩy.

4. Dấy nháy đơn ('') để chỉ chuỗi rỗng.

**20**

**BÀI 4**

**SỬ DỤNG BIẾN TRONG CHƯƠNG TRÌNH**

***Biến là gì?***

***Cách sử dụng biến trong chương trình.***

**1. Biến là công cụ trong lập trình**

Hoạt động cơ bản của chương trình máy tính là xử lí dữ liệu. Trước khi được máy tính xử lí, mọi dữ liệu nhập vào đều được lưu trong bộ nhớ của máy tính. Ví dụ, nếu muốn cộng hai số a và b, trước hết hai số đó sẽ được nhập và lưu trong bộ nhớ máy tính, sau đó máy tính sẽ thực hiện phép cộng a + b.

Để chương trình luôn biết chính xác dữ liệu cần xử lí được lưu ở vị trí nào trong bộ nhớ, các ngôn ngữ lập trình cung cấp một công cụ lập trình rất quan trọng. Đó là ***biến nhớ***, hay được gọi ngắn gọn là ***biến***.

***Trong lập trình, biến được dùng để lưu trữ dữ liệu và dữ liệu được biến lưu trữ có thể thay đổi trong khi thực hiện chương trình.***

Dữ liệu do biến lưu trữ được gọi là ***giá trị của biến***.

Chúng ta hãy xét một số ví dụ để hiểu vai trò của biến nhớ trong lập trình.

Trong bài thực hành 2, em đã biết, để có kết quả của phép cộng 15 + 5 và in ra màn hình, em có thể sử dụng câu lệnh Python sau đây:

Câu hỏi đặt ra ở đây là làm thế nào để in ra màn hình tổng của hai số mà giá trị của chúng không biết trước (các số là kết quả của một quá trình tính toán trung gian nào đó). Bằng cách sử dụng hai biến X, Y để lưu giá trị của các số đó, câu lệnh sau đây sẽ in ra màn hình giá trị tổng của chúng:

Hình dưới đây minh họa trực quan việc lưu trữ các số 15 và 5 trong các ô nhớ có “tên” tương ứng là **X** và **Y** mà chương trình sẽ lấy ra để thực hiện phép cộng.

**21**

***Ví dụ 1***: Giả sử cần tính giá trị của các biểu thức 10050 và 10050 , sau đó in

kết quả ra màn hình. Chúng ta có thể tính các biểu thức này một cách trực tiếp. Để ý rằng tử số trong các biểu thức là như nhau. Do đó có thể tính giá trị tử số và lưu tạm thời trong một biến trung gian **X**, sau đó thực hiện các phép chia, về mặt toán học, điểu này được thực hiện như sau:

**2. Khai báo biến**

***Tất cả các biến trong chương trình cần phải được khai báo trước khi sử dụng đến.***

Việc khai báo biến gồm: Khai báo *tên biến*.

Khai báo *kiểu dữ liệu* của biến.

*Tên biến* phải tuân theo quy tắc đặt tên của ngôn ngữ lập trình. ***Ví dụ 2:*** Cách khai báo biến trong ngôn ngữ Python:

Tên biến = giá trị khởi tạo.

Chẳng hạn, khai báo các biến cùng giá trị khởi tạo ban đầu trong python.

Khai báo

X = 5 Y = 0.0 x = X

lop = '8A3'

dayso = [1,3,5,7,9]

Giải thích

Biến *X* có giá trị khởi tạo là 5, thuộc kiểu **int** Biến *Y* có giá trị khởi tạo là 0.0, thuộc kiểu **float**

Biến ***x*** có giá trị khởi tại là giá trị *X*, kiểu dữ liệu của *X* Biến *lop* có giá trị khởi tạo là 8A3, thuộc kiểu **str**

Biến *dayso* có giá trị khởi tạo là một tập hợp 5 phần tử, mỗi phần tử là kiểu **int**, *dayso* thuộc kiểu **list**

***Lưu ý:*** *Python phân biệt chữ cái in hoa và chữ cái thường, cho phép tên biến là tiếng Việt có dấu*. Chẳng hạn, các tên biến: ***gia, Gia, GIa, GIA, giá, Giá, …*** là các biến khác nhau hoàn toàn. Hình dưới đây minh họa sự khác nhau đó:

Kết quả in ra màn hình là:

**22**

Thông thường các ngôn ngữ lập trình sẽ yêu cầu khai báo biến đi kèm với kiểu dữ liệu của biến ở phần khai báo của chương trình và kiểu dữ liệu có *tính tĩnh* trong suốt quá trình thực thi, nhưng Python thì cho phép khai báo biến tùyý ở trong chương trình, không cần phải khai báo kèm kiểu dữ liệu, chương trình tự nhận diện kiểu dữ liệu theo giá trị mà nó được khởi tạo, *kiểu dữ liệu của biến là động*, kể cả trong các định nghĩa hàm hoặc cấu trúc lặp.

Hình dưới đây cho thấy *tính động* kiểu dữ liệu của biến, biến thay đổi kiểu dữ liệu tính từ khi nó được nhận giá trị mới:

Lệnh **type**(*biến*) để kiểm tra kiểu dữ liệu hiện tại của *biến* trong python. Tùy theo ngôn ngữ lập trình, cú pháp khai báo biến có thể khác nhau.

**3. Sử dụng biến trong chương trình**

Sau khi khai báo, ta có thể sử dụng các biến trong các câu lệnh để tính toán hoặc xử lí chúng như với các giá trị dữ liệu (số, kí tự hay xâu, …). Điều phải lưu ý là để có các kết quả tính toán đúng mục tiêu của chương trình, cần phải gán các giá trị dữ liệu thích hợp cho các biến.

Như vậy các thao tác có thể thực hiện với các biến là: *Gán* giá trị cho biến;

*Tính toán* với các biến.

*Khi được gán một giá trị mới, giá trị cũ của biến bị xoá đi.* Ta có thể thực hiện việc gán giá trị cho biến tại bất kì thời điểm nào trong chưong trình. Hay *giá trị của biến có thể thay đổi*.

Tuỳ theo ngôn ngữ lập trình, kí hiệu của câu lệnh gán cũng có thể khác nhau. Ví dụ, trong ngôn ngữ Python, người ta kí hiệu phép gán là dấu bằng (**=**), phân biệt với phép so sánh bằng là hai dấu bằng (**==**).

***Vídụ3:***Bảngdướiđâymôtảlệnhgángiá trịvà tínhtoánvớicác biếntrongPython:

**Lệnh trong Python** X = 12

X = Y

X = (a+b)/2

X = X + 1

**Ý nghĩa** Gán giá trị số 12 vào biến nhớ *X.*

Gán giá trị đã lưu trong biến nhớ Y vào biến nhớ *X.* Thực hiện phép toán tính trung bình cộng hai giá trị nằm

trong hai biến nhớ *a* và *b.* Kết quả gán vào biến nhớ *X.* Tăng giá trị của biến nhớ *X* lên 1 đơn vị, kết quả gán trở

lại biến *X.* (cách viết khác: X +=1).

**23**

Trong Python, giá trị của biến còn có thể gán nhờ câu lệnh nhập dữ liệu **input()**. ***Ví dụ 4:*** Nhập giá trị cho các biến **m**, **n** bằng lệnh input() trong Python.

Khi gặp các câu lệnh trên trong chương trình, máy tính sẽ đợi người dùng gõ các giá trị tương ứng của các biến **m**, **n** từ bàn phím và nhấn phím **Enter**.

CÂU HỎI VÀ BÀI TẬP

1.Giả sử **Y** được khai báo là biến với kiểu dữ liệu số thực, **X** là biến với kiểu dữ liệu xâu. Các phép gán sau đây có hợp lệ không?

A) Y = 4 B) X = 3242 C) X = '3242' D) Y = "Lam Dong"

2. Muốn nhập dữliệutừ bàn phímcho biến m. Câu lệnh nào sao đâylà đúng trong Python? A) m = print() B) m = 3242 C) m = input() D) input(n)

3. Hãy cho biết kiểu dữ liệu của các biến cần khai báo dùng để viết chưong trình để giải các bài toán dưới đây:

a) Tính diện tích S của hình tam giác với độ dài một cạnh a và chiều cao tương ứng h (a và h là các số tự nhiên được nhập vào từ bàn phím).

b) Tính kết qủa c của phép chia lấy phần nguyên và kết qủa d của phép chia lấy phần dư của hai số nguyên a và b.

**24**

**BÀI THỰC HÀNH**

**3**

**KHAI BÁO VÀ SỬ DỤNG BIẾN**

***Bước đầu làm quen cách khai báo và sử dụng biến trong chương trình.***

**NỘI DUNG**

Hãy xem lại các kiểu dữ liệu trong Python nêu trong bài 3 để thực hành cách khai báo biến với các kiểu dữ liệu khác nhau.

Cú pháp khai báo biến trong Python:

**< *danh sách các biến* > = < *danh sách các giá trị khởi tạo ban đầu* >**

trong đó:

- ***danh sách các biến*** là danh sách *một* hoặc *nhiều* tên biến được cách nhau bởi dấu phẩy.

- ***danh sách các giá trị khởi tạo ban đầu*** là giá trị lần lượt được gán cho các biến để chương trình xác định kiểu dữ liệu cho biến tương ứng.

***Bài 1:* Tìm hiểu một số cách khai báo biến trong Python:**

Cho đoạn khai báo biến trong ngôn ngữ Python như sau:

a) Điền vào bảng sau:

Tên biến Giá trị khởi tạo Kiểu dữ liệu

**25**

b) Khởi động Python và gõ chính xác đoạn chương trình sau:

c) Lưu lại với tên ***Bai1TH3.py***, dịch và chạy chương trình. Hãy nhập từ bàn phím xâu “**Nguyễn Văn Tèo**” và nhấn phím **Enter**. Quan sát kết quả in ra màn hình.

d) Nêu các cách khai báo biến mà em nhận biết được từ đoạn chương trình trên.

e) Lưu lại bản sao với tên file ***Bai1TH3e.py***, sửa lại các câu lệnh như hình dưới. Chạy chương trình và nêu nhận xét của em?

***Bài 2:* Viết chương trình Python có khai báo và sử dụng biến:**

***Bài toán:*** Một cửa hàng cung cấp dịch vụ bán hàng thanh toán tại nhà. Khách hàng chỉ cần đăng kí số lượng mặt hàng cần mua, nhân viên cửa hàng sẽ giao hàng và nhận tiền thanh toán tại nhà khách hàng. Ngoài trị giá hàng hoá, khách hàng còn phải trả thêm phí dịch vụ. Hãy viết chương trình Python để tính tiền thanh toán trong trường hợp khách hàng chỉ mua một mặt hàng duy nhất.

***Gợi ý:*** Công thức cần tính:

***Tiền thanh toán = Đơn giá*** x ***Số lượng + Phí dịch vụ***

a) Khởi động Python. Gõ chương trình sau và tìm hiểu ý nghĩa của từng câu lệnh trong chương trình:

**26**

b) Lưu chương trình với tên ***Bai2TH3.py***. Dịch và chạy chương trình. Hãy nhập vào bộ dữ liệu (đơn giá và số lượng). Khi nhập xong, chương trình sẽ báo lỗi như hình dưới:

***Tìm hiểu và sửa lỗi:*** Hãy chú ý đến hai vị trí chính được đánh dấu trong đoạn báo lỗi trên. Ta tạm dịch thông báo lỗi như sau: “tại dòng 8, thanhtien = soluong\*dongia + phi. ***Lỗi kiểu dữ liệu***: *không thể thực hiện phép toán số của kiểu xâu*”.

Giải thích lỗi: *Trong ngôn ngữ Python quy định các giá trị được nhập vào từ bàn phím bằng lệnh input() đều có kiểu dữ liệu mặc định là xâu*. Để chương trình nhận đó là giá trị số thì phải chỉ định kiểu số mà chương trình chuyển đổi như sau:

**int**(input())

**float**(input())

Giá trị nhập vào chuyển thành số nguyên

Giá trị nhập vào chuyển thành số thực

c) Hãy sửa lại các dòng lệnh như sau, dịch và chạy chương trình với bộ dữ liệu (1000; 20). Quan sát kết quả thực hiện của chương trình.

d) Chạy chương trình với các bộ dữ liệu khác. Quan sát kết quả nhận được.

e) Chạy chương trình và thử nhập bộ dữ liệu (15.3; 25) (giả sử *đơn giá* tính bằng USD). Hãy tìm hiểu lỗi và đề xuất cách sửa lỗi?

f) Sửa lại hai câu lệnh như hình sau, dịch và chạychương trình với bộ dữ liệu (15.3; 20). Quan sát kết quả thực hiện của chương trình và rút ra nhận xét.

Hihi! Hãy tìm hiểu thêm **eval()** nhé!

**27**

***Bài 3:*** Thử viết chương trình nhập hai số X và Y từ bàn phím. Sau đó hoán đổi các giá trị của X và Y rồi in ra màn hình giá trị của X và Y. Lưu với tên

Tham khảo chương trình sau:

Ơ! dấu **;** là sao? choáng hết cả đầu.các bạn ơi!

**TỔNG KẾT**

1. Cú pháp khai báo biến trong Python:

**< *danh sách các biến* > = < *danh sách các giá trị khởi tạo ban đầu* >** trong đó:

- ***danh sách các biến*** là danh sách *một* hoặc *nhiều* tên biến được cách nhau bởi dấu phẩy.

- ***danh sách các giá trị khởi tạo ban đầu*** là giá trị lần lượt được gán cho các biến để chương trình xác định kiểu dữ liệu cho biến tương ứng. 2. Kí hiệu **=** được sử dụng trong lệnh gán giá trị cho biến.

3. Các lệnh **int(input()), float(input()), eval(input())** dùng chuyển giá trị nhập từ bàn phím thành số tương ứng.

4. Dấy **#** để viết chú thích giúp chương trình dễ đọc, dễ hiểu. Chương trình sẽ bỏ qua không dịch dòng này.

5. Dấu ; để viết các lệnh *riêng biệt*, *cùng cấp* trên cùng một dòng.

**28**

**BÀI 5**

**TỪ BÀI TOÁN ĐẾN CHƯƠNG TRÌNH**

***Khái niệm về bài toán và xác định bài toán.***

***Quá trình giải bài toán trên máy tính.*** ***Thuật toán và cách thức mô tả thuật toán.***

**1. Xác định bài toán** Như vậy, có thể hiểu:

***Bài toán là một công việc hay một nhiệm vụ cần phải giải quyết.***

Để giải quyết được một bài toán cụ thể, người ta cần *xác định bài toán*, tức là phát biểu rõ *các điều kiện cho trước* và *kết quả cần thu được*.

***Ví dụ 1.*** Xét các bài toán tính diện tích hình tam giác, tìm đường đi tránh các điểm nút giao thông trong giờ cao điểm và nấu một món ăn.

a) Để tính diện tích hình tam giác:

*Điều kiện cho trước*: Một cạnh và đường cao tương ứng với cạnh đó; *Kết quả cần thu được*: Diện tích hình tam giác.

b) Đối với bài toán tìm đường đi tránh các điểm nút giao thông trong giờ cao điểm:

*Điều kiện cho trước*: Vị trí điểm nghẽn giao thông và các con đường có thể đi từ vị trí hiện tại tới vị trí cần tới;

*Kết quả cần thu được*: Đường đi từ vị trí hiện tại tới vị trí cần tới mà không qua điểm nghẽn giao thông.

c) Đối với bài toán nấu một món ăn:

*Điều kiện cho trước*: Các thực phẩm hiện có (trứng, mỡ, mắm, muối, rau,...); *Kết quả cần thu được*: Một món ăn.

***Xác định bài toán là bước đầu tiên và là bước rất quan trọng trong việc giải bài toán.***

**29**

**2. Quá trình giải bài toán trên máy tính**

Máy tính chỉ có thể thực hiện các công việc tiếp nhận, xử lí, biến đổi, tính toán, lưu

trữ và biểu diễn thông tin thành dạng cần thiết dưới sự chỉ dẫn của con người thông qua các câu lệnh cụ thể.

***Việc dùng máy tính giải một bài toán chính là đưa cho máy tính dày hữu hạn các thao tác đơn giản mà nó có thể thực hiện được để từ các điều kiện cho trước ta nhận được kết quả cần tìm.***

Dãy hữu hạn các thao tác cần thực hiện để giải một bài toán được gọi là thuật toán. Máy tính không thể tự mình tìm ra lời giải của các bài toán. Cách giải của một bài toán cụ thể, tức thuật toán, là *tư duy sáng tạo của con người*. Tuy nhiên, việc mô tả thuật toán chưa đủ đối với máy tính mà cần diễn đạt thuật toán dưới dạng mà máy tính có thể hiểu và thực hiện được. *Kết quả diễn đạt thuật toán là chương trình được viết trong một ngôn ngữ lập trình nào đó*. Máy tính sẽ chạy chương trình và cho ta lời giải của bài toán.

Nói một cách khác, thuật toán là các bước để giải một bài toán, còn chương trình là thể hiện của thuật toán trong một ngôn ngữ lập trình cụ thể.

Như vậy, quá trình giải bài toán trên máy tính gồm các bước sau:

***Xác định bài toán***: Từ phát biểu của bài toán, ta xác định đâu là ***Điều kiện cho trước*** - **thông tin đã cho** (**INPUT**) và đâu là kết quả cần nhận được – **thông tin cần tìm** (**OUTPUT**).

***Mô tả thuật toán***: Diễn tả cách giải bài toán bằng dãy các thao tác cần phải thực hiện.

***Viết chương trình***: Dựa vào thuật toán ở trên, viết chương trình bằng một ngôn ngữ lập trình thích hợp.

Cần phải lưu ý rằng, để giải một bài toán có thể có nhiều thuật toán khác nhau, song mỗi thuật toán chỉ dùng để giải một bài toán cụ thể. Vì vậy, khi mô tả thuật toán, người ta thường *chỉ ra cả* điều kiện cho trước và kết quả cần nhận được kèm theo để dễ nhận biết thuật toán đó dùng để giải bài toán nào.

**3. Thuật toán và mô tả thuật toán**

Trong phần này chúng ta sẽ tìm hiểu sâu hon về khái niệm thuật toán.

**30**

Nhiều công việc chúng ta thường làm mà không phải suy nghĩ nhiều, tuy nhiên, nếu hệ thống lại, ta có thể thấy thực chất đó là những thuật toán. Đon giản như việc pha trà mời khách có thể mô tả dưới dạng thuật toán như sau:

INPUT: Trà, nước sôi, ấm và chén. OUTPUT: Chén trà dã pha để mời khách.

*Bước 1*. Tráng ấm, chén bằng nước sôi. *Bước 2.* Cho trà vào ấm.

*Bước 3.* Rót nước sôi vào ấm và đợi khoảng 3 đến 4 phút. *Bước 4.* Rót trà ra chén để mời khách.

Việc *liệt kê* các bước như trên là một cách thường dùng để *mô tả thuật toán.* Nếu không có mô tả gì khác trong thuật toán, các bước của thuật toán được thực hiện một cách tuần tự theo trình tự như đã được chỉ ra.

Mặc dù không được nêu rõ trong khái niệm thuật toán, song thuật toán phải được mô tả đủ cụ thể để bất kì đối tượng nào, với cùng khả năng và điều kiện như nhau, khi thực hiện thuật toán cũng đều đạt dược kết quả như nhau. Để minh hoạ, chúng ta xét thêm một vài ví dụ:

***Bài toán*** *“Giải phương trình bậc nhất dạng tổng quát ax + b =* 0”: INPUT: Các số a và b*.*

OUTPUT: Nghiệm của phương trình bậc nhất. *Bước 1.* ***Nếu*** a = 0 ***thì*** chuyển tới bước 3.

*Bước 2.* Tính nghiệm của phương trình *x*  *a* và chuyển tới bước 4. *Bước 3.* ***Nếu*** *b ≠ 0* ***thì*** thông báo phương trình đã cho vô nghiệm.

Ngược lại (b = 0) ***thì*** thông báo phương trình có vô số nghiệm. *Bước 4.* Kết thúc.

***Bài toán*** *“Làm món trứng tráng”* INPUT: Trứng, dầu ăn, muối và hành. OUTPUT: Trứng tráng.

*Bước 1.* Đập trứng, tách vỏ và cho trứng vào bát.

*Bước 2.* Cho một chút muối và hành tươi thái nhỏ vào bát trứng. Dùng đũa khuấy mạnh để trộn trứng, muối và hành.

*Bước 3.* Cho một thìa dầu ăn vào chảo, đun nóng dầu rồi đổ trứng vào. Đun tiếp khoảng 1 phút.

*Bước 4.* Lật mặt trên của miếng trứng úp xuống dưới. Đun tiếp khoảng 1 phút.

*Bước 5.* Lấy trứng ra đĩa.

Rõ ràng, bất kì ai biết về các phép toán số học hay hiểu biết một chút vể làm bếp, theo đúng trình tự và chỉ dẫn ở các bước trong các thuật toán nêu trên đều có thể tính ra

**31**

nghiệm của phương trình đã cho hay tự làm cho mình món trứng tráng. Tóm lại, có thể hiểu:

***Thuật toán là dày hữu hạn các thao tác cần thực hiện theo một trình tự xác định để thu được kết quả cần thiết từ những điểu kiện cho trước.***

**4. Một số ví dụ về thuật toán**

***Ví dụ 2.*** Một hình A được ghép từ một hình chữ nhật với chiều rộng 2a, chiều dài b và một hình bán nguyệt bán kính a như hình dưới đây:

INPUT: Các số *a* là 2 chiều rộng của hình chữ nhật và là bán kính của hình bán nguyệt, *b* là chiều dài của hình chữ nhật.

OUTPUT: Diện tích của hình *A.*

Thuật toán đơn giản để tính diện tích hình *A* có thể gồm các bước sau: *Bước* 1. *S* 2*a**b* (Tính diện tích hình chữ nhật).

*Bước 2. S*2 22 (Tính diện tích hình bán nguyệt). *Bước 3. S* *S*1 *S*2 và kết thúc.

*Lưu* ý: Trong biểu diễn thuật toán, người ta cũng thường sử dụng kí hiệu  để chỉ phép gán giá trị của một biểu thức cho một biến.

***Ví dụ 3.*** Tính tổng của 100 số tự nhiên đầu tiên. INPUT: Dãy 100 số tự nhiên đầu tiên: 1, 2,…,100. OUTPUT: Giá trị của tổng 1 + 2 + ...+ 100.

Ở đây ta sẽ tính trực tiếp tổng cần tìm mà không sử dụng công thức toán ho5cba82ng cách dùng một biến SUM để lưu giá trị của tổng. Việc tính SUM có thể được thực hiện như sau: Đầu tiên gán cho SUM giá trị bằng 0; tiếp theo lần lượt thêm các giá trị 1, 2, 3, ..., 100 vào SUM. Vấn đề là ở chỗ tổ chức việc “lần lượt thêm vào” như thế nào? Cách dễ nhận thấy nhất là thực hiện liên tiếp 100 phép cộng:

*Bước 1.* SUM  0.

*Bước 2.* SUM  SUM + 1. *…*

*Bước 101.* SUM  SUM + 100.

Tuy nhiên, việc mô tả thuật toán như trên là quá dài dòng (nhất là khi không chỉ tính tổng của 100 số mà số các số cần tính tổng lớn hơn nhiều). Đê ý một chút ta có thể thấy trong tất cả các bước nêu trên đều chỉ có một phép toán được thực hiện: cộng thêm vào SUM lần lượt các giá trị 1, 2, 3,..., 100. Tức là chỉ có một thao tác “*cộng*” được lặp

**32**

đi lặp lại 100 lần. Mặt khác, việc cộng thêm số ***i*** vào SUM chỉ được thực hiện khi ***i*** không vượt quá 100. Vì vậy, thuật toán tìm SUM có thể dược mồ tả ngắn gọn hon như sau:

*Bước 1.* SUM  0; ***i***1.

*Bước 2.* SUM SUM + ***i***; ***i******i*** + 1.

*Bước 3.* Nếu I ≤ 100 thì quay lại bước 2. Ngược lại, thông báo giá trị SUM và kết thúc thuật toán.

***Ví dụ 4.*** Đổi giá trị của hai biến x và y*.*

INPUT: Hai biến x, y có giá trị tương ứng là  và *.* OUTPUT: Hai biến x, y có giá trị tương ứng là  và *.*

Trong bài 3 của Bài thực hành 3, chúng ta đã tìm hiểu và viết chương trình hoán đổi các giá trị của hai biến X và Y*.* Ví dụ này sẽ mô tả thuật toán để viết chương trình đó.

Ta không thể thực hiện trực tiếp hai phép gán: xy và yx, bởi sau phép gán thứ nhất, giá trị của x đã bị thay bằng giá trị của y và kết quả của hai phép gán này là cả hai biến x và y cùng có giá trị ban đầu của biến y*.* Vì thế, cần dùng một biến trung gian, ví dụ biến z, để lưu tạm thời giá trị của biến x. Do vậy, ta có thuật toán sau:

*Bước 1*. zx (Sau bước này giá trị của *z* sẽ bằng ) *Bước 2.* xy (Sau bước này giá trị của x sẽ bằng ) *;*

*Bước 3.* yz (Sau bước này giá trị của y sẽ bằng giá trị của z, chính là , giá trị ban đầu của biến x)

***Ví dụ 5.*** Cho hai số thực a và b. Hãy cho biết kết quả so sánh hai số đó dưới dạng “a lớn hơn b”, “a nhỏ hơn b" hoặc "a bằng b".

INPUT: Hai số thực a và *b.* OUTPUT: Kết quả so sánh.

**33**

Bài toán rất đơn giản. Thoạt đầu ta thấy thuật toán sau đây có vẻ như có thể giải quyết bài toán này:

*Bước 1.* So sánh a và b. Nếu *a > b,* kết quả là “a lớn hơn b”.

*Bước 2.* Nếu *a < b,* kết quả là “a nhỏ hơn b”; ngược lại, kết quả là “a bằng b” và kết thúc thuật toán.

Tuy nhiên, nếu thử lại các bước với *a* = 6 và *b* = 5, ta sẽ thấy sau bước 1 có kết quả *“a* lớn hơn *b”,* nhưng đến bước 2, khi kiểm tra *a < b* không thoả mãn ta lại có tiếp kết quả “a bằng *b”* và như thế ta nhận được hai kết quả.

Vì vậy, để có kết quả đúng, cần mô tả chính xác hơn điều kiện kết thúc thuật toán như sau:

*Bước 1*. Nếu *a > b,* kết quả là “a lớn hơn *b*” và *chuyển đến bước* 3.

*Bước 2.* Nếu *a < b,* kết quả là “a nhỏ hơn b”; Ngược lại, kết quả là “a bằng b”*. Bước 3.* Kết thúc thuật toán.

***Ví dụ 6.*** Tìm số lớn nhất trong dãy *A* các số a1, a2, ..., *an* cho trước.

Ta sẽ dùng biến *MAX* để lưu giá trị phần tử lớn nhất của dãy *A.* Việc xác định *MAX* có thể được thực hiện như sau: Đầu tiên gán giá trị a1 cho biến *MAX.* Tiếp theo, lần lượt so sánh các số a2,…, *an* của dãy *A* với *MAX.* Nếu ai > *MAX,* ta gán *a*i cho *MAX.*

INPUT: Dãy *A* các số a1, a2, ..., *an (n* ≥1). OUTPUT: Giá trị *MAX =* max{a1, a2, ..., *an*}*.* Từ đó, ta có thuật toán sau:

*Bước 1. MAX*a1; i1.

*Bước 2. Nếu ai > MAX, gán MAX**ai. Bước 3. i**i* + 1.

*Bước 4. Nếu i ≤ n, quay lại bước 2.*

*Bước 5.* Thông báo giá trị *MAX* và kết thúc thuật toán.

Dưới đây minh hoạ thuật toán trên với trường hợp chọn thỏ nặng nhất trong bốn chú thỏ có trọng lượng tương ứng là 2, 1, 5, 3 ki-lô-gam.

a) Trước hết, ta gán *MAX* là trọng lượng của thỏ số 1, tức *MAX* = 2.

b) So sánh *MAX* với trọng lượng của thỏ số 2. Vì trọng lượng của thỏ số 2 nhỏ hơn *MAX*, do đó *MAX* vẫn bằng 2.

c) Tiếp theo, so sánh *MAX* với trọng lượng của thỏ số 3. MAX = 2 Vì trọng lượng của thỏ số 3 lớn hơn *MAX,* do đó *MAX* được đặt

lại bằng 5.

MAX = 5

d) Cuối cùng, so sánh *MAX* với trọng lượng của thỏ số 4. *MAX* lớn hơn trọng lượng của thỏ số 4, do đó *MAX* vẫn bằng 5. Kết quả, thỏ nặng nhất có trọng lượng là 5kg.

MAX = 5

CÂU HỎI VÀ BÀI TẬP

1. Hãy chỉ ra INPUT và OUTPUT của các bài toán sau:

a) Xác định số học sinh trong lớp cùng mang họ Trần.

b) Tính tổng của các phần tử lớn hơn 0 trong dày n số cho trước. c) Tìm số các số có giá trị nhỏ nhất trong n số đà cho.

2. Giả sử x và y là các biến số. Hãy cho biết kết quả của việc thực hiện thuật toán sau: *Bước 1.* x x + y

*Bước 2.* y x - y *Bước 3.* x x - y

3. Cho trước ba số dương a, b và c. Hãy mô tả thuật toán cho biết ba số đó có thể là độ dài ba cạnh của một tam giác hay không.

4. Cho hai biến x và y. Hãy mô tả thuật toán đổi giá trị của các biến nói trên (nếu cần) để x và y theo thứ tự có giá trị không giảm.

5. Hãy cho biết kết quả của thuật toán sau: *Bước 1.* SUM  0; I  0.

*Bước 2.* Nếu i > 100 thì chuyển tới bước 4.

*Bước 3.* i  i+1; SUM  SUM + i. Quay lại bước 2. *Bước 4.* Thông báo giá trị SUM và kết thúc thuật toán.

6. Hãy mô tả thuật toán giài bài toán tính tổng các số dương của dãy số A ={a1, a2, …, an} cho trước.

**35**

**BÀI 6** **CÂU LỆNH ĐIỀU KIỆN**

***Cấu trúc rẽ nhánh và hai dạng cấu trúc rẽ nhánh.***

***Câu lệnh điều kiện thể hiện cấu trúc rẽ nhánh.***

**1. Hoạt động phụ thuộc vào điều kiện**

Trong các ví dụ trên liên quan tới việc phải điều chỉnh hành động tùy theo hoàn

cảnh cụ thể, ta thấy từ "nếu" được dùng để chỉ một "'điều kiện" tương ứng với hoàn cảnh đó. Các điều kiện đó là: “Em bị ốm” hoặc “Trời mưa”. Hoạt động tiếp theo của em hoặc của bạn Long sẽ phụ thuộc vào các điếu kiện đó có được thỏa mãn hay không, hay nói cách khác, hoạt động tiếp theo phụ thuộc vào kết quả kiểm tra điều kiện đưa ra đúng hay sai.

**Điều kiện** **Kiểm tra** **Kết quả** **Hoạt động tiếp theo**

Long nhìn ra ngoài trời

và thấy trời mưa.

Buổi sáng thức dậy, em Em bị ốm? thấy mình hoàn toàn

khoẻ mạnh.

Đúng

Sai

Long ở nhà (không đi đá bóng)

Em tập thể dục buổi sáng như thường lệ.

Khi kết quả kiểm tra là *đúng,* ta nói điều kiện được *thoả mãn,* còn khi kết quả kiểm tra là *sai,* ta nói điều kiện *không thoả màn.*

Ngoài những điều kiện gắn với các sự kiện đời thường như trên, trong Tin học chúng ta có thể gặp nhiều dạng điều kiện khác, ví dụ:

*Nếu* nháy nút ở góc trên, bên phải cửa sổ trên màn hình máy tính, *(thì)* cửa sổ **36**

sẽ được đóng lại.

*Nếu X* > 5, *(thì*) in giá trị của *X* ra màn hình.

*Nếu* nhấn phím Pause/Break, *(thì)* chương trình (sẽ bị) ngừng. **2. Điều kiện và phép so sánh**

Để so sánh hai giá trị số hoặc hai biểu thức có giá trị số, chúng ta sử dụng các kí hiệu toán học như: =, ≠, <, ≤, > và ≥. Chúng ta cũng đã biết rằng các phép so sánh có kết quả *đúng* hoặc *sai.*

Các phép so sánh có vai trò rất quan trọng trong việc mô tả thuật toán và lập trình. Chúng thường được sử dụng để biểu diễn các điều kiện.

***Phép so sánh cho kết quả đúng có nghĩa điều kiện được thoả mãn; ngược lại, điều kiện không được thoả mãn.***

***Ví dụ 1.*** Ta muốn chương trình in ra màn hình giá trị lớn hơn trong số hai giá trị của các biến *a* và *b.* Khi đó giá trị của biến *a* hoặc *b* được in ra phụ thuộc vào phép so sánh *a > b* là đúng hay sai:

*“Nếu a > b, in giá trị của biến a ra màn hình; ngược lại, in giá trị của biến b ra màn hình.”*

Trong trường hợp này điều kiện được biểu diễn bằng phép so sánh *a > b.*

Tương tự, khi giải phương trình bậc nhất dạng tổng quát *ax + b* = 0, để tính nghiệm của phương trình chúng ta cần kiểm tra các điều kiện được cho bằng các phép so sánh *a* = 0 và *c ≠* 0.

**3. Cấu trúc rẽ nhánh**

Nói chung, khi thực hiện chương trình, máy tính sẽ *thực hiện tuần tự* các câu lệnh, từ trên xuống dưới. Để thay đổi trình tự ấy, ngôn ngữ lập trình có các câu lệnh cho phép máy tính thực hiện một câu lệnh nào đó, nếu một điều kiện cụ thể được thoả mãn; ngược lại, nếu điều kiện không được thoả mãn thì bỏ qua câu lệnh hoặc thực hiện một câu lệnh khác.

***Ví dụ 2.*** Một hiệu sách thực hiện đợt khuyến mãi lớn với nội dung sau: Nếu mua sách với tổng số tiền ít nhất là 100 nghìn đồng, khách hàng sẽ được giảm 30% tổng số tiền phải thanh toán. Hãy mô tả hoạt động tính tiền cho khách.

Ta có thể mô tả việc tính tiền cho khách hàng bằng các bước dưới đây: *Bước 1.* Tính tổng số tiền *T* khách hàng đã mua sách.

*Bước 2.* Nếu *T ≥* 100000, số tiền phải thanh toán là 70%×*T*. *Bước 3.* In hoá đơn.

***Ví dụ 3.*** Cũng như trong ví dụ 2, nhưng chính sách khuyến mãi được thực hiện như sau: Nếu tổng số tiền từ 100 nghìn đồng trở lên, khách hàng sẽ được giảm 30% tổng số tiền phải thanh toán. Trong trường hợp ngược lại, những khách hàng mua với tổng số tiền không đến 100 nghìn đồng sẽ chỉ được giảm 10%.

Khi đó, cần phải tính lại tiền cho khách trong cả hai trường hợp, tổng tiền không nhỏ hơn 100 nghìn đồng và tổng tiền nhỏ hơn 100 nghìn đồng. Thuật toán có thể được

**37**

sửa lại như sau:

*Bước 1.* Tính tổng số tiền *T* khách hàng đã mua sách.

*Bước 2.* Nếu *T* ≥ 100000, số tiền phải thanh toán là 70%×*T*; Ngược lại, số tiền phải thanh toán là 90%×*T*.

*Bước 3.* In hoá đơn.

Cách thể hiện hoạt động phụ thuộc vào điều kiện như trong Ví dụ 2 được gọi là *cấu trúc rẽ nhánh dạng thiếu*, còn trong ví dụ 3 được gọi là *cấu trúc rẽ nhảnh dạng đủ*.

***Cấu trúc rẽ nhánh cho phép thay đổi thứ tự thực hiện tuần tự các bước trong thuật toán.***

Cấu trúc rẽ nhánh giúp cho việc lập trình được linh hoạt hơn. **4. Câu lệnh điều kiện**

***Trongcác ngôn ngữlậptrình,các cấu trúcrẽnhánh được thểhiện bằng câu lệnh điều kiện.***

Trong Python, ***câu lệnh điều kiện dạng thiếu*** được viết với từ khoá **if** như sau: **if** *<điều kiện>***:**

**<*câu lệnh*>**

Khi gặp *câu lệnh điều kiện* này, chưong trình sẽ kiểm tra *điều kiện.* Nếu *điều kiện* được thỏa mãn, chương trình sẽ thực hiện ***câu lệnh*** sau dấu hai chấm (**:**). Ngược lại, ***câu lệnh*** đó bị bỏ qua.

***Ví dụ 4.*** Giả sử cần in ra màn hình số lớn hơn trong hai số a và b: ***Nếu a > b thì*** in ra màn hình giá trị của a.

Thể hiện bằng câu lệnh điều kiện dạng thiếu trong Python:

***Ví dụ 5.*** Viết chương trình yêu cầu người dùng nhập một số không lớn hơn 5 từ **38**

bàn phím, chương trình sẽ kiểm tra tính hợp lệ, nếu không hợp lệ sẽ thông báo lỗi. Khi đó, chương trình có thể biểu diễn bằng thuật toán sau:

*Bước 1.* Nhập số a.

*Bước 2. Nếu a >* 5 *thì* thông báo lỗi.

Thể hiện bằng câu lệnh điều kiện dạng thiếu trong Python như sau:

Trong Python, ***câu lệnh điều kiện dạng đủ*** được viết với từ khoá **if** và **else** như sau:

**if** *<điều kiện>***: <*câu lệnh 1*>**

**else:**

**<*câu lệnh 2*>**

Với câu lệnh điều kiện này, chương trình sẽ kiểm tra *điều kiện.* Nếu *điều kiện* được thoả mãn, chương trình sẽ thực hiện ***câu lệnh 1***. Trong trường hợp ngược lại, ***câu lệnh 2*** sẽ được thực hiện.

***Chú ý:*** Trong Python, ***câu lệnh*** sau dấu hai chấm (**:**) có thể viết cùng dòng hoặc viết xuống dòng thì phải thụt vào một số khoảng trắng so với dòng lệnh chứa dấu hai chấm, quy định như sau:

**Câu lệnh sau dấu hai chấm trong Python**

***Lệnh đơn*** (chỉ có một câu lệnh)

***Lệnh ghép*** (có từ hai câulệnhtrở lênvàcùng cấp)

**Cách viết**

hoặc

hoặc

***Ví dụ 6.*** Viết chương trình tính kết quả của *a* chia cho *b,* với *a* và *b* là hai số bất kì. Phép tính chỉ thực hiện được khi *b ≠ 0*. Chương trình sẽ kiểm tra giá trị của *b,* nếu *b ≠ 0* thì thực hiện phép chia; nếu *b* = 0 sẽ thông báo lỗi.

*Nếu b ≠ 0 thì* tính kết quả, *ngược lại thi* thông báo lỗi.

Dưới đây là câu lệnh Python thể hiện cấu trúc rẽ nhánh dạng đủ nói trên:

**39**

CÂU HỎI VÀ BÀI TẬP

1. Em hãy nêu một vài ví dụ về các hoạt động hằng ngày phụ thuộc vào điều kiện. 2. Mỗi điều kiện hoặc biểu thức sau cho kết quả đúng hay sai?

a) 123 là số chia hết cho 3.

b) Nếu ba cạnh a, b và c của một tam giác thỏa mãn c2 > a2 + b2 thì tam giác đó có một góc vuông.

c) 152 > 200. d) x2 < 1.

3. Hai người bạn cùng chơi trò đoán số. Một người nghĩ trong đầu một số tự nhiên nhỏ hơn 10. Người kia đoán xem bạn đã nghĩ số gì. Nếu đoán đúng, người đoán sẽ được cộng thêm 1 điểm, nếu sai sẽ không được cộng diểm. Luân phiên nhau nghĩ và đoán. Sau 10 lần, ai đưọc nhiều điểm hơn, người đó sẽ thắng.

Hãy phát biểu quy tắc thực hiện một nước đi ở trò chơi. Hoạt động nào sẽ được thực hiện, nếu điều kiện của quy tắc đó thỏa mãn? Hoạt động nào sẽ được thực hiện, nếu điều kiện của quy tắc đó không thỏa mãn?

4. Các câu lệnh Python sau dây được viết đúng hay sai? a) if x = 7: a = b

b) if x > 5: a= b

c) if x > 5: a = b; m = n; d) if x > 5: a= b else m= n

5. Với mỗi câu lệnh sau đây giá trị của biến x sẽ là bao nhiêu, nếu trước đó giá trị của x bằng 5?

a) if (45%3) == 0: x = x + 1 b) if x > 10: x = x + 1

6. Giả sử cần viết chương trình nhập một số tự nhiên vào máy tính và in ra màn hình kết quả số đã nhập là số chẵn hay lẻ, chẳng hạn “*5 là số lẻ*”, “*8 là số chẵn*”. Hãy mô tả các bước của thuật toán để giải quyết bài toán trên và viết chương trình Python để thực hiện thuật toán đó.

**40**

**BÀI THỰC HÀNH**

**4**

**SỬ DỤNG CÂU LỆNH ĐIỀU KIỆN**

***- Luyện tập sử dụng câu lệnh điều kiện if.***

***- Rèn luyện kĩ năng ban đầu về đọc các chương trình đơn giản và hiểu được ý nghĩa của thuật toán sử dụng trong chương trình.***

**NỘI DUNG**

***Bài 1:*** Viết chương trình nhập hai số nguyên *a* và *b* khác nhau từ bàn phím và in hai số đó ra màn hình theo thứ tự không giảm.

a) Mô tả thuật toán để giải bài toán đã cho.

b) Gõ chương trình sau đây, lưu với tên *Bai1TH4.py*:

c) Tìm hiểu ý nghĩa của các câu lệnh trong chương trình. Nhấn F5 để dịch và sửa lỗi gõ, nếu có. Chạy chương trình với các bộ dữ liệu (12, 53), (65, 20) để thử chương trình.

***Bài 2:*** Viết chương trình nhập chiều cao của hai bạn Long và Trang, in ra màn hình kết quả so sánh chiều cao của hai bạn, chẳng hạn “Bạn Long cao hơn”.

a) Mô tả thuật toán để giải bài toán đã cho.

b) Gõ chương trình sau đây, lưu với tên *Bai2TH4.py*:

c) Chạy chương trình với các bộ dữ liệu (1.5, 1.6), (1.6, 1.5) và (1.6, 1.6). Quan sát các kết quả nhận được và nhận xét. Hãy tìm chỗ chưa đúng trong chương trình.

d) Sửa lại chương trình để có kết quả đúng: chỉ in ra màn hình một thông báo kết quả.

**41**

Tham khảo và tìm hiểu 3 đoạn chương trình sau đây:

Dùng 3 lệnh điều kiện **if** dạng thiếu.

1

Dùng lệnh **if** lồng nhau.

2

Cách viết gọn **if** lồng nhau bằng từ khóa **elif**. Cú pháp: **if** <điều kiện 1>**:**

*<Câu lệnh 1>*

3 **elif** <điều kiện 2>**:** *<Câu lệnh 2>*

**…. else:**

*<Câu lệnh n>*

e) Hãynhận xét về số lần trình biên dịch phải thực hiện ở 3 đoạn chương trình trên?

***Bài 3:*** Dưới đây là chưong trình nhập ba số dương a, *b* và *c* từ bàn phím, kiểm tra và in ra màn hình kết quả kiểm tra ba số đó có thể là độ dài các cạnh của một tam giác hay không.

***Ý tưởng*:** Ba số dưong *a, b* và *c* là độ dài các cạnh của một tam giác khi và chỉ khi *a + b > c, b + c > a* và c + *a > b.*

a) Gõ chương trình sau đây, lưu với tên *Bai3TH4.py*:

b) Tìm hiểu ý nghĩa của các câu lệnh trong chưong trình, dịch và chạy chưong trình với các số tuỳ ý.

**42**

*Lưu ý.* Trong chưong trình trên chúng ta sử dụng từ khoá and để kết hợp nhiều phép so sánh đơn giản thành một phép so sánh phức hợp. Giá trị của phép so sánh này là *đúng*

*tất cả* các phép so sánh đơn giản đểu có giá trị *đúng.* Ngược lại, chỉ cần ánh thành phần có giá trị *sai* thì nó có giá trị *sai.*

1. Câu lệnh điều kiện dạng thiếu: **if** *<điều kiện>***:**

**<*câu lệnh*>** 2. Câu lệnh điều kiện dạng đủ:

**if** *<điều kiện>***: <*câu lệnh 1*>**

**else:**

**<*câu lệnh 2*>**

3. Câu lệnh điều kiện rẽ nhánh: **if** <điều kiện 1>**:**

*<Câu lệnh 1>* **elif** <điều kiện 2>**:**

*<Câu lệnh 2>*

**…. else:**

*<Câu lệnh n>*

4. Có thể sử dụng các câu lệnh **if** lồng nhau.

5. Sử dụng từ khoá **and** có thể kết hợp nhiều phép so sánh đơn giản thành một phép so sánh phức hợp. Giá trị của phép so sánh này là *đúng* khi và chỉ khi *tất cả* các phép so sánh đơn giản đều *đúng.* Ngược lại, nó có giá trị *sai.*

Ví dụ: (a > 0) **and** (a <= 5)

Từ khoá **or** cũng được sử dụng để kết hợp nhiều phép so sánh đơn giản. Giá trị của phép so sánh này chỉ *sai* khi *tất cả* các phép so sánh thành phần đều *sai.* Ngược lại, nó có giá trị *đúng.*

Ví dụ: (a > 0) **or** (a <= 5)

**43**

**BÀI 7 CÂU LỆNH LẶP** ***Cấu trúc lặp.***

***Câu lệnh lặp for thể hiện cấu trúc lặp với số lần lặp cho trước.***

**1. Câu lệnh lặp – một lệnh thay cho nhiều lệnh**

***Ví dụ 1.*** Giả sử ta cần vẽ ba hình vuông có cạnh 1 đơn vị. Mỗi hình vuông là ảnh dịch chuyển của hình bên trái nó một khoảng cách 2 đơn vị. Do đó, ta chỉ cần lặp lại thao tác vẽ hình vuông ba lần. Việc vẽ hình có thể thực hiện được bằng thuật toán sau đây:

*Bước 1.* Vẽ hình vuông (vẽ liên tiếp bốn cạnh và trở về đỉnh ban đầu).

*Bước 2.* Nếu số hình vuông đã vẽ được ít hơn 3, di chuyển bút vẽ về bên phải 2 đơn vị và trở lại bước 1; ngược lại, kết thúc thuật toán.

Riêng với bài toán vẽ một hình vuông***,*** thao tác chính là vẽ bốn cạnh bằng nhau, hay lặp lại bốn lần thao tác vẽ một đoạn thẳng. Sau mỗi lần vẽ đoạn thẳng, thước kẻ được quay một góc 90° sang phải tại vị trí của bút vẽ. Thuật toán sau đây sẽ mô tả các bước để vẽ hình vuông:

**44**

*Bước 1.* Đặt k  0 (k là số đoạn thẳng đã vẽ được).

*Bước 2.* Vẽ đoạn thẳng độ dài bằng 1 đơn vị và quay thước 900 sang phải. kk+1. *Bước 3.* Nếu k < 4, trở lại bước 2; Ngược lại, kết thúc thuật toán.

Lưu ý rằng, biến k được sử dụng như là biến đếm để ghi lại số cạnh đã vẽ được. ***Ví dụ 2.*** Giả sử cần tính tổng của 100 số tự nhiên đầu tiên, tức là tính:

S = 1 + 2 + 3 + ... + 100.

Hoạt động chính khi giải bài toán này là thực hiện phép cộng. Thuật toán trong ví dụ 3, bài 5 đã mô tả việc thực hiện lặp lại phép cộng 100 lần.

Cách mô tả các hoạt động lặp trong thuật toán như trong ví dụ trên được gọi là *cấu trúc lặp.*

Mọi ngôn ngữ lập trình đều có các “cách” để chỉ thị cho máy tính thực hiện cấu trúc lặp với một câu lệnh. Đó là các *câu lệnh lặp*.

**2. Câu lệnh lặp for**

Các ngôn ngữ lập trình thường có nhiều dạng câu lệnh lặp. Một trong các câu lệnh lặp đơn giản trong Python là câu lệnh lặp biết trước số lần lặp, có dạng:

**for** <*biến đếm*> **in range(***giá trị đầu*, *giá trị cuối***): <*câu lệnh*>**

trong đó: **for**, **in**, **range** là các từ khóa.

Trong bài này, chúng ta chỉ xét: ***giá trị đầu*** nhỏ hơn ***giá trị cuối*** và là các giá trị nguyên. Vì vậy, <*biến đếm*> sẽ là biến kiểu số nguyên.

Câu lệnh sẽ thực hiện nhiều lần, mỗi lần là một vòng lặp. Số vòng lặp là biết trước và bằng (***giá trị cuối* – *giá trị đầu***).

Khi thực hiện, ban đầu *biến đếm* sẽ nhận ***giá trị đầu***, sau mỗi vòng lặp, *biến đếm* tự động tăng thêm một đơn vị cho đến khi bằng (***giá trị cuối* – *1***) thì dừng lặp.

***Ví dụ 3.*** Chương trình sau in ra màn hình thứ tự lần lặp và giá trị biến đếm:

***Ví dụ 4.*** Chương trình sau in liên tiếp 10 dấu \* trên một dòng màn hình.

Trong đoạn chương trình trên, ***giá trị đầu*** bằng 0. Chỉ dẫn để lệnh **print()** sau khi in ra dấu **\*** thì kết thúc lệnh bằng cách viết tiếp ký tự đặt trong cặp dấu nháy '', mặc định thì kết thúc sẽ xuống một dòng màn hình.

Trong thực tế, để có mười kết quả, chúng ta phải thực hiện mười lần một hoạt động (có thể với những điều kiện khác nhau). Máy tính thực hiện công việc xử lí thông tin thay cho con người và cũng phải thực hiện ngần ấy hoạt động. Câu lệnh lặp góp phần giúp giảm nhẹ công sức viết chương trình máy tính.

**45**

**3. Tính tổng và tích bằng câu lệnh lặp**

***Ví dụ 5.*** Chương trình sau đây sẽ tính tổng của N số tự nhiên đầu tiên, với N là số tự nhiên được nhập vào từ bàn phím.

***Ví dụ 6.*** Ta kí hiệu N! là tích N số tự nhiên đầu tiên, đọc là N giai thừa: N! = 1.2.3…N

Dưới đây là chương trình tính N! với N là số tự nhiên được nhập vào từ bàn phím. Chương trình sử dụng câu lệnh lặp **for**.

CÂU HỎI VÀ BÀI TẬP

1. Cho một vài ví dụ về hoạt động được thực hiện lặp lại trong cuộc sống hằng ngày. 2. Chương trình Python sau đây thực hiện hoạt động nào.

3. Hãy mô tả thuật toán để tính tổng A sau đây, n là số tự nhiên được nhập vào từ bàn phím. Thử viết chương trình Python tính A?

𝐴 = 1.3 + 2.4 + 3.5 + ⋯+ 𝑛(𝑛 + 2)

**46**

**BÀI THỰC HÀNH**

**5**

**SỬ DỤNG LỆNH LẶP for**

***-Viết chương trình Python có sử dụng câu lệnh lặp for.***

***- Tiếp tục nâng cao kĩ năng đọc và tìm hiểu chương trình.***

**NỘI DUNG**

***Bài 1.*** Viết chương trình in ra màn hình bản cửu chương của số N trong khoảng từ 1 đến 9, số N được nhập từ bàn phím.

a) Gõ chương trình sau đây, lưu với tên *Bai1TH5.py*

b) Tìm hiểu ý nghĩa của các câu lệnh trong chương trình, dịch và sửa lỗi chương trình (nếu có).

c) Chạy chương trình với các giá trị N được nhập vào lần lượt bằng 1, 2, …, 9. Quan sát kết quả nhận được trên màn hình.

***Bài 2.*** Chỉnh sửa chương trình để làm đẹp kết quả trên màn hình.

Kết quả của chương trình nhận được trong bài 1 có hai nhược điểm sau đây: - Các hàng kết quả quá sát nhau nên khó đọc.

- Các hàng kết quả không được cân đối với hàng tiêu để.

Nên sửa chương trình bằng cách chèn thêm một hàng trống giữa các hàng kết quả và đẩy các hàng này sang phải một khoảng cách nào đó.

a) Chỉnh sửa câu lệnh lặp của chương trình như sau:

**47**

b) Dịch và chương trình với các giá trị gõ vào từ bàn phím. Quan sát kết quả nhận được trên màn hình.

***Bài 3.*** Cũng như câu lệnh **if**, có thể dùng câu lệnh **for** lồng bên trong một câu lệnh for khác khi thực hiện lặp. Sử dụng các câu lệnh **for** lồng nhau để in ra màn hình các số từ 0 đến 99 theo dạng bảng như hình sau:

a) Gõ chương trình sau đây, lưu với tên *Bai3TH5.py*.

b) Dịch, sửa lỗi và chạy chương trình. Quan sát kết quả trên màn hình. Chỉ dẫn là cách ra một tab (là 1 khoảng trắng bằng nhau).

**TỔNG KẾT**

1. Câu lệnh lặp với số lần biết trước trong Python có dạng: **for** <*biến đếm*> **in range(***giá trị đầu*, *giá trị cuối***):**

**<*câu lệnh*>**

2. Có thể sử dụng các câu lệnh **for** lồng nhau. Khi đó các *biến đếm* trong các câu lệnh **for** phải khác nhau.

**48**

**LẶP VỚI SỐ LẦN CHƯA**

**BÀI 8** **XÁC ĐỊNH TRƯỚC**

***Cấu trúc lặp với số lần lặp không xác định trước.***

***Câu lệnh lặp với số lần chưa biết trước while.***

*Bước 1.* Nhập số n từ bàn phím.

*Bước 2.* Nếu n < 5 quay trở về bước 1. *Bước 3.* …

**1. Lệnh lặp với số lần chưa biết trước**

***Ví dụ 1.*** Nếu cộng lần lượt n số tự nhiên đầu tiên (n = 1, 2, 3,...), ta sẽ được các kết quả T1 = 1, T2 =1 + 2, T3 =1 + 2 + 3,... tăng dần. Cần cộng bao nhiêu số tự nhiên đầu tiên để ta nhận được tổng Tn nhỏ nhất lớn hơn 1000? Trong trường hợp này, để quyết định thực hiện phép cộng với số tiếp theo hay dừng, trong từng bước cần phải kiểm tra tổng đã lớn hơn 1000 hay chưa.

Chúng ta hãy tìm hiểu các bước của thuật toán trong ví dụ này một cách cụ thể hơn. Kí hiệu S là tổng cần tìm và ta có thuật toán như sau:

*Bước 1.* S  0, n  0.

*Bước 2.* Nếu S ≤ 1000 thì chuyển tới bước 3; Ngược lại (S > 1000) chuyển tới bước 4. *Bước 3. n*  *n* + 1; S  S + n; và quay lại bước 2.

**49**

*Bước 4.* In kết quả: S và n là số tự nhiên nhỏ nhất sao cho S > 1000. Kết thúc.

Việc thực hiện phép cộng ở thuật toán trên được lặp lại với số lần chưa xác định trước, phụ thuộc vào một điều kiện (S ≤ 1 000) và chỉ dừng khi kết quả kiểm tra điều kiện đó sai (S > 1 000) .

Để viết chương trình chỉ dẫn máy tính thực hiện các hoạt động lặp mà chưa xác định trước được số lần lặp, ta có thể sử dụng câu lệnh có dạng lặp với số lần chưa xác định.

Nói chung các ngôn ngữ lập trình đều có câu lệnh lặp dạng này. Cụ thể, câu lệnh lặp với số lần chưa xác định trước trong Python có dạng:

**while** *<điều kiện>***: <*câu lệnh*>**

trong đó:

*<điều kiện>* thường là một phép so sánh.

**<*câu lệnh*>** có thể là câu lệnh đơn hay câu lệnh ghép. Câu lệnh lặp này được thực hiện như sau:

*Bước 1.* Kiểm tra *điều kiện.*

*Bước 2.* Nếu *điều kiện* sai, *câu lệnh* sẽ bị bỏ qua và việc thực hiện lệnh lặp kết thúc. Nếu *điều kiện* đúng, thực hiện ***câu lệnh*** và quay lại bước 1.

***Chú ý:*** Nếu *<điều kiện> luôn luôn* đúng trong mọi trường hợp thì lệnh lặp **while** không thể kết thúc. Khi đó chương trình “*rơi*” vào “*vòng lặp vô tận*”, đây là lỗi lập trình cần tránh.

**2. Ví dụ lặp với số lần chưa biết trước**

***Ví dụ 2.*** Chương trình nhập vào một số n từ bàn phím cho đến khi số được nhập vào lớn hơn 100 thì kết thúc.

***Ví dụ 3.*** Chúng ta biết rằng, nếu n (n > 0) càng lớn thì 𝑛 càng nhỏ, nhưng luôn lớn hơn 0. Với giá trị nào của n thì 𝑛 < 0.005 hoặc 𝑛 < 0.003?

Chương trình dưới đây tìm số n nhỏ nhất để 𝑛 nhỏ hơn một sai số cho trước.

Chạy chương trình ta sẽ nhận được kết quả n = 334 (sai số là 0.00299). Lần lượt thay giá trị biến saiso = 0.002, saiso = 0.001 ta nhận được các kết quả n = 501 và n =

1001. Em hãy kiểm tra các kết quả này bằng cách tính lại phép chia 𝑛.

**50**

***Ví dụ 4.*** Chương trình dưới đây thể hiện thuật toán tính số n trong Ví dụ 1:

Khi chạy chương trình ta nhận được kết quả in ra màn hình là:

***Ví dụ 5.*** Để viết chương trình tính tổng 𝑇 = 1 + 2 + 3 + ⋯ + 100. Hai đoạn chương trình dưới đây đều cho củng một kết quả.

Ví dụ này cho thấy rằng chúng ta có thể sử dụng câu lệnh **while** thay cho câu lệnh **for**.

CÂU HỎI VÀ BÀI TẬP

1. Nêu một vài ví dụ về hoạt động lặp với số lần chưa biết trước.

2. Hãy phát biểu sự khác biệt giữa câu lệnh lặp với số lần cho trước và câu lệnh lặp với số lần chưa xác định.

3. Hãy tìm hiểu các thuật toán sau đây và cho biết khi thực hiện thuật toán, máy tính sẽ thực hiện bao nhiêu vòng lặp? Khi kết thúc, giá trị của S bàng bao nhiêu? Viết chưong trình Python thể hiện các thuật toán đó.

a) Thuật toán 1

*Bước 1.* S  10, x  0.5.

*Bước 2.* Nếu S ≤ 5.2, chuyển tới bước 4. *Bước 3.* S  S – x và quay lại bước 2.

*Bước 4.*

b) Thuật toán 2

*Bước 1.* S  10, n  0.

*Bước 2.* Nếu S ≥ 10, chuyển tới bước 4. *Bước 3.*

*Bước 4.*

4. Hãy tìm hiểu mỗi đoạn chương trình Python sau đây và cho biết với đoạn lệnh đó chương trình thực hiện bao nhiêu vòng lặp. Hãy rút ra nhận xét của em.

a) Chương trình 1 b) Chương trình 2

**51**

**BÀI THỰC HÀNH**

**6**

**SỬ DỤNG LỆNH LẶP while**

***- Viết chương trình Python sử dụng câu lệnh lặp với số lần không xác định trước.***

***- Rèn luyện khả năng đọc chương trình, tìm hiểu tác dụng của các câu lệnh.***

**NỘI DUNG**

***Bài 1:*** Viết chương trình sử dụng lệnh lặp **while** để tính trung bình n số thực xl, x2, x3,..., xn. Các số n và xl, x2, x3,..., xn được nhập vào từ bàn phím.

***Ỷ tưởng:*** Sử dụng một biến đếm và lệnh lặp **while** để nhập và cộng dần các số vào một biến kiểu số thực cho đến khi nhập đủ n số.

a) Mô tả thuật toán của chương trình, các biến dự định sẽ sử dụng và kiểu của chúng. b) Gõ chương trình Python sau đây và lưu chương trình với tên *Bai1TH6.py*:

c) Đọc và tìm hiểu ý nghĩa của từng câu lệnh. Dịch chương trình và sửa lỗi, nếu có. Chạy chương trình với các bộ dữ liệu được gõ từ bàn phím và kiểm tra kết quả nhận được.

d) Viết lại chương trình bằng cách sử dụng câu lệnh **for** thay cho câu lệnh **while**. Tham khảo đoạn chương trình dưới đây:

***Bài 2:*** Tìm hiểu chương trình nhận biết một số tự niên N được nhập vào từ bàn phím có phải là số nguyên tố hay không.

***Ỷ tưởng:*** Kiểm tra lần lượt N có chia hết cho các số tự nhiên 2 ≤ i ≤ N – 1 hay không. Kiểm tra tính chia hết bằng phép chia lấy phần dư (%).

a) Đọc và tìm hiểu ý nghĩa của từng câu lệnh trong chương trình sau đây:

**52**

b) Gõ, dịch và chạy thử chương trình với một vài độ chính xác khác nhau.

**TỔNG KẾT**

Câu lệnh lặp với số lần chưa xác định trong Python có dạng: **while** *<điều kiện>***:**

**<*câu lệnh*>**

**53**

**BÀI 9** **LÀM VIỆC VỚI DÃY SỐ** ***Dữ liệu kiểu danh sách.***

*L****àm việc với biến danh sách.***

***Sử dụng các biến kiểu mảng và câu lệnh lặp.***

**1. Dãy số và biến mảng**

Do tại mỗi thời điểm một biến chỉ lưu được một giá trị duy nhất nên trong đoạn chương trình trên, mỗi khi cần tới thu nhập của hộ gia đình nào ta lại phải thực hiện câu lệnh **a =** eval**(**input**(**'Thu nhập của gia đình thứ '**+**str**(i) +** ': '**))** để nhập mức thu nhập của hộ đó vào biến **a**. Cần lưu ý thêm, thao tác nhập mức thu nhập của các hộ gia đình từ bàn phím chiếm phần lớn thời gian trong quá trình thực hiện đoạn chương trình trên, mà ta lại phải thực hiện công việc đó hai lần.

Để chỉ phải nhập dữ liệu một lần, ta có thể khai báo nhiều biến, mỗi biến dùng để lưu trữ thu nhập của một hộ gia đình.

**54**

Ví dụ, trong Python ta cần nhiều câu lệnh khai báo và nhập dữ liệu như sau: thunhap\_0 **=** eval**(**input**(**'Thu nhập của gia đình thứ 0: '**))**

thunhap\_1 **=** eval**(**input**(**'Thu nhập của gia đình thứ 1: '**))** …

thunhap\_4 **=** eval**(**input**(**'Thu nhập của gia đình thứ 4: '**))**

Chú ý rằng địa phương cần khảo sát có bao nhiêu hộ gia đinh thì cần viết đủ chừng ấy khai báo và câu lệnh nhập mức thu nhập - một công việc không hề thú vị.

**Dữ liệu *kiểu mảng* là *một tập hợp hữu hạn các phần tử có thứ tự, mọi phần tử đều có chung một kiểu dữ liệu,* gọi là kiểu của phần tử. Việc sắp thứ tự được thực hiện bằng cách gán cho mỗi phần tử một *chỉ số*.**

Hình dưới, minh họa mảng chứa dữ liệu của 5 hộ gia đình và gán chỉ số theo thứ tự từ 0 đến 4.

Các phần tử có thể có cùng kiểu dữ liệu bất kì. *Trong bài này, chúng ta chỉ xét các mảng có các phần tử kiểu* ***số nguyên*** *hoặc* ***số thực*** *và gọi là* ***các dãy số.***

Khi khai báo một biến có kiểu dữ liệu là kiểu mảng, biến đó được gọỉ là *biến mảng.* Có thể nói rằng, khi sử dụng biến mảng, về thực chất chúng ta sắp xếp theo chỉ số các biến có *cùng kiểu* dưới một tên duy nhất.

Giá trị của biến mảng là một *mảng,* tức ***một dãy số*** (số nguyên, hoặc số thực) có thứ tự, mỗi số là giá trị của biến thành phần tương ứng.

**2. Ví dụ về biến mảng**

Để làm việc với các dãy số nguyên hay số thực, chúng ta phải khai báo ***biến mảng*** có kiểu tương ứng trong chương trình.

Cách khai báo ***biến mảng*** trong các ngôn ngữ lập trình có thể khác nhau, nhưng luôn cần chỉ rõ: *tên biến mảng, số lượng phần tử, kiểu dữ liệu chung* của các phần tử.

Ví dụ, cách khai báo đơn giản một biến mảng trong ngôn ngữ Python như sau: tên biến mảng **= [**giá trị khởi tạo**]**\*số lượng phần tử

Chẳng hạn: a **= [**0**]**\*5  mảng a có 5 phần tử kiểu số nguyên. b **= [**0.0**]**\*10  mảng b có 10 phần tử kiểu số thực.

***Lưu ý quan trọng:*** Trong Python, ***chỉ số*** của một mảng *mặc định từ số 0 trở đi*.

***Ví dụ 1.*** Tiếp tục với ví dụ ở mục 1, thay vì khai báo các biến thunhap\_0, thunhap\_1, thunhap\_2,... để lưu mức thu nhập của các hộ gia đình, ta khai báo biến mảng thunhap như sau trong Python:

**55**

Cách khai báo và sử dụng biến mảng như trên có lợi gì?

Trước hết, có thể thay rất nhiều câu lệnh nhập và in dữ liệu ra màn hình bằng một câu lệnh lặp. Chẳng hạn, ta có thể viết:

để nhập mức thu nhập của các hộ gia đình. Thay vì phải viết 5 câu lệnh khai báo và 5 câu lệnh nhập, ta chỉ cần viết hai câu lệnh là đủ và kết quả đạt được là như nhau.

Ta còn có thể sử dụng biến mảng một cách rất hiệu quả trong xử lí dữ liệu. Để so sánh mức thu nhập của các hộ gia đình với một giá trị nào đó, ta cũng chỉ cần một câu lệnh lặp, chẳng hạn:

Điều này giúp tiết kiệm rất nhiều thời gian và công sức viết chương trình.

Sau khi một mảng đã được khai báo, chúng ta có thể làm việc với các phần tử của nó như làm việc với một biến thông thường như gán giá trị, đọc giá trị và thực hiện các tính toán với các giá trị đó thông qua *tên của biến mảng* và *chỉ số* tương ứng của phần tử. Chẳng hạn, trong các câu lệnh sau **thunhap[i]** là phần tử thứ *i* của biến mảng **thunhap***.*

Ta có thể gán giá trị cho các phần tử của mảng **thunhap** bằng câu lệnh gán: thunhap[0] = 5000000

thunhap[2] = 8000000

hoặc nhập dữ liệu từ bàn phím bằng câu lệnh lặp:

Viết lại đoạn chương trình ở trên như sau:

**56**

**3. Tìm giá trị lớn nhất và nhỏ nhất của dãy số**

Viết chương trình nhập *N* số nguyên từ bàn phím và in ra màn hình số nhỏ nhất và số lớn nhất. *N* cũng được nhập từ bàn phím.

Trước hết ta khai báo biến *N* để nhập số các số nguyên sẽ được nhập vào. Sau đó khai báo *N* biến lưu các số được nhập vào như là các phần tử của một biến mảng *A.* Ngoài ra, cần khai báo một biến *Max* để lưu số lớn nhất, *Min* để lưu số nhỏ nhất.

Chương trình được viết bằng Python như sau:

Trong chương trình này, chúng ta hãy lưu ý: Số phần tử của mảng A được khai báo phụ thuộc vào biến N được nhập vào từ bàn phím.

CÂU HỎI VÀ BÀI TẬP

1. Hãy nêu các lợi ích của việc sư dụng biến mảng trong chương trình.

2. “Có thể xem biến mảng là một biến được tạo từ nhiều biến có cùng kiểu, nhưng chỉ có một tên duy nhất". Phát biểu đó đúng hay sai?

3. Viết chương trình sử dụng biến mảng để nhập từ bàn phím các phần từ của một dãy số. Độ dài của dãy cũng được nhập từ bàn phím.

**57**

**BÀI THỰC HÀNH**

**7**

***- Làm quen với việc khai báo và sử dụng các biến mảng.***

***- Ôn luyện cách sử dụng câu lệnh for.***

***- Củng cố kĩ năng đọc, hiểu và chỉnh sửa chương trình.***

**NỘI DUNG**

***Bài 1:*** Viết chương trình nhập điểm của các bạn trong lớp. Sau đó in ra màn hình số bạn đạt kết quả học tập loại giỏi, khá, trung bình và kém (theo tiêu chuẩn từ 8.0 trở lên đạt loại giỏi, từ 6.5 đến 7.9 đạt loại khá, từ 5.0 đến 6.4 đạt loại trung bình và dưới 5.0 xếp loại kém).

a) Xem lại về cách sử dụng và khai báo biến mảng trong Python. b) Liệt kê các biến dự định sẽ sử dụng trong chương trình.

c) Khởi động Python, gõ chương trình sau vào máytính và lưu với tên *Bai1TH7.py*. Tìm hiểu các câu lệnh của chương trình. Dịch và chạy thử chương trình.

***Bài 2:*** Viết chương trình nhập vào một dãy số nguyên gồm N phần tử. N được nhập từ bàn phím. In ra màn hình dãy số theo thứ tự tăng dần.

Tham khảo đoạn chương trình sau:

**TỔNG KẾT**

1. Cách khai báo đơn giản một biến mảng trong ngôn ngữ Python như sau:

tên biến mảng **= [**giá trị khởi tạo**]**\*số lượng phần tử

2. Tham chiếu tới phần tử của mảng được xác định bằng cách:

<tên biến mảng>[chỉ số] **58**