

NỘI DUNG ÔN TẬP CUỐI HỌC KÌ 2 MÔN TIN HỌC 10

BÀI 6: CÂU LỆNH Rẽ NHÁNH

1. Cấu trúc rẽ nhánh trong mô tả thuật toán

Các ngôn ngữ lập trình bậc cao cung cấp công cụ mô tả <điều kiện>, tính giá trị <điều kiện> và thể hiện cấu trúc rẽ nhánh dựa trên giá trị tính được của <điều kiện>.

2. Điều kiện rẽ nhánh

- Trong mô tả thuật toán, <điều kiện> rẽ nhánh là một biểu thức logic True hoặc False. Bảng 1. Kí hiệu phép so sánh trong Python

So sánh	Kí hiệu trong Python
Lớn hơn	>
Lớn hơn hoặc bằng	>=
Nhỏ hơn	<
Nhỏ hơn hoặc bằng	<=
Bằng	=
Khác	!=

Kết nối các biểu thức logic với nhau bằng các phép tính logic (and – và, or – hoặc, not – phủ định) ta lại nhận được một biểu thức logic

Phép tính	Biểu thức	Ý nghĩa
and	x and y	Cho kết quả True khi và chỉ khi x và y nhận True
or	x or y	Cho kết quả False khi và chỉ khi x và y nhận False
not	not x	Đảo giá trị logic x

3. Câu lệnh rẽ nhánh trong chương trình Python

- Python cung cấp hai câu lệnh rẽ nhánh:

+ *Câu lệnh rẽ nhánh dạng if*

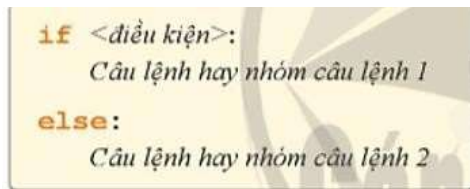
```
if <điều kiện>:  
    Câu lệnh hay nhóm câu lệnh
```

Ví dụ: Minh họa chương trình sử dụng câu lệnh if trong

Python Hình 6.4: Chương trình kiểm tra số nguyên dương

có hai chữ số

+ *Câu lệnh rẽ nhánh if – else:*



Hình 6.5: Cách viết của câu lệnh **if-else**

Câu lệnh hoặc các câu lệnh cùng nhóm viết lùi vào trong một số vị trí so với dòng điều kiện và viết thẳng hàng với nhau gọi là một khối lệnh.

Ví dụ:

Lưu ý: Cách viết các câu lệnh trong Python:

- Các câu lệnh ở khối trong viết lùi các đầu dòng nhiều hơn các lệnh khối ngoài.
- Các câu lệnh cùng một khối có khoảng cách đầu dòng như nhau.

BÀI 8: CÂU LỆNH LẶP

1. Cấu trúc lặp trong mô tả thuật toán

- Khi có một thao tác cần được thực hiện lặp lại một số lần liên tiếp trong quá trình thực hiện thuật toán thì cần dùng cấu trúc lặp.

Có hai kiểu cấu trúc lặp:

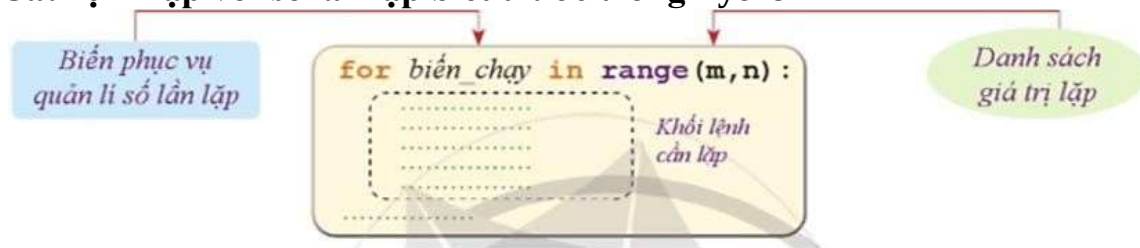
- Thuật toán biết trước số lần lặp.

Ví dụ: Thuật toán của việc in ra màn hình máy tính 10 dòng “Xin chào Python”.

- Thuật toán không biết trước số lần lặp.

Ví dụ: Khi mô tả thuật toán cho máy tính hỏi và kiểm tra mật khẩu thì ta không tính trước được số lần máy tính yêu cầu nhập lại mật khẩu, vì chừng nào mật khẩu nhập vào chưa đúng thì máy tính còn hỏi lại.

2. Câu lệnh lặp với số lần lặp biết trước trong Python



Hình 8.1: Cấu trúc câu lệnh lặp dạng **for**

Trong câu lệnh **for**, hàm `range(m,n)` dùng để khởi tạo dãy số nguyên từ `m` đến `n-1` (với `m`

`< n`). Trường hợp `m = 0`, `range(m, n)` viết gọn là `range(n)`.

Ví dụ: Minh họa một câu lệnh **for** trong Python và kết quả thực hiện.



Hình 8.2: Ví dụ câu lệnh **for**

Ví dụ:

3. Câu lệnh lặp với số lần lặp không biết trước trong Python

Trong Python, câu lệnh lặp với số lần không biết trước có dạng:

```
while <điều kiện>:
    Câu lệnh hay nhóm câu lệnh
```

BÀI 10: CHƯƠNG TRÌNH CON VÀ THƯ VIỆN CÁC CHƯƠNG TRÌNH CON CÓ SẴN

1. Khái niệm chương trình con

- Khi lập trình để giải bài toán có thể chia bài toán đó thành các chương trình con, viết các đoạn chương trình giải các bài toán con.
 - Ngôn ngữ lập trình bậc cao cho phép người lập trình tạo ra chương trình con bằng cách đặt tên một đoạn chương trình gồm các câu lệnh thực hiện việc nào đó.
- ⇒ Sử dụng các chương trình con là một trong những cách giúp việc lập trình trở nên dễ dàng hơn.

2. Khai báo và gọi thực hiện một hàm trong Python

- Có thể gọi một chương trình con trong Python là một hàm. Để sử dụng hàm cần khai báo hàm và viết lời gọi thực hiện.
- Hàm trong Python được khai báo theo mẫu sau:

```
def tên_hàm (tham số):
    Các lệnh mô tả hàm
```

Trong đó:

- + Tên hàm phải đặt theo quy tắc đặt tên trong Python.
- + Theo sau tên hàm có thể có hoặc không có các tham số.
- + Phần thân hàm (gồm các lệnh mô tả hàm) phải viết lùi vào theo quy định của Python.

3. Chuyển dữ liệu cho hàm thực hiện

Một hàm có thể thực hiện với những giá trị do chương trình truyền vào qua lời gọi hàm, tương ứng với danh sách tham số. Có hai cách truyền dữ liệu cho hàm thực hiện:

- *Cách thứ nhất*, chương trình gọi thực hiện với các giá trị cụ thể.
- *Cách thứ hai*, chương trình gọi thực hiện hàm với giá trị tham số truyền vào.

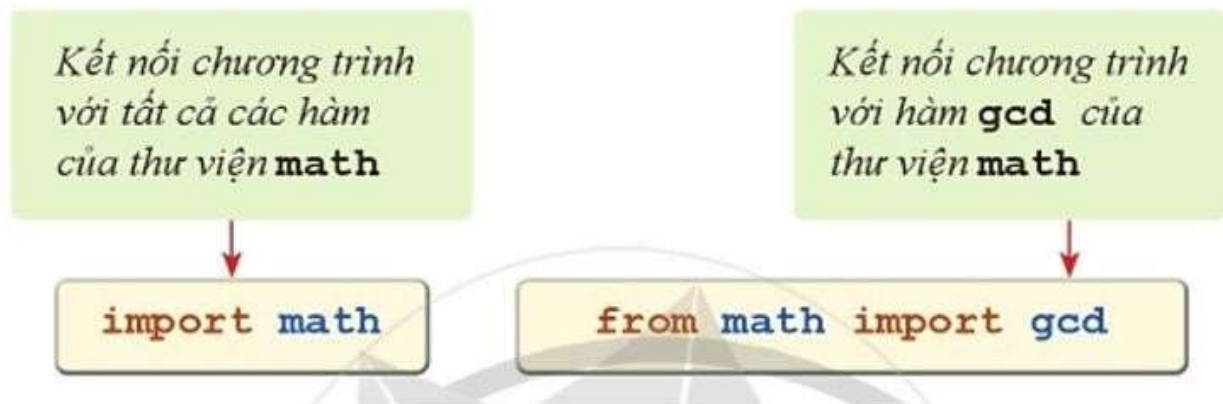
4. Lời gọi hàm

- Trong nhiều ngôn ngữ lập trình bậc cao, hàm trả về chương trình một giá trị qua tên của nó, như vậy hàm được sử dụng như một biến trong chương trình gọi nó.
- Trong Python, một hàm có thể trả về một giá trị qua tên của nó nếu như có lệnh **return**

return
<Giá trị> trước khi ra khỏi hàm.

5. Các hàm được xây dựng sẵn

- Mỗi tập hợp gồm một số hàm được xây dựng sẵn thường gọi là một thư viện.
- Trong chương trình, người lập trình chỉ cần gọi hàm có sẵn, thực hiện mà không cần phải tự xây dựng lại hàm.
- Để có thể sử dụng các hàm trong thư viện cần kết nối thư viện hoặc hàm đó với



chương trình.

BÀI 12: KIỂU DỮ LIỆU XÂU KÍ TỰ - XỬ LÝ XÂU KÍ TỰ

1. Kiểu dữ liệu chuỗi ký tự

- Một chuỗi ký tự là một dãy các ký tự, trong Python chuỗi ký tự được đặt trong cặp nháy kép (“...”) hoặc nháy đơn (‘...’).
- Các ký tự trong chuỗi được đánh số bắt đầu từ 0. Python cung cấp hàm **len()** để đếm ký tự trong chuỗi kể cả ký tự dấu cách, số ký tự trong chuỗi được gọi là độ dài của chuỗi.

2. Một số hàm xử lý chuỗi ký tự

a) Ghép chuỗi bằng phép +

Viết liên tiếp các chuỗi cần ghép theo thứ tự và đặt giữa hai chuỗi kề nhau dấu “+”.

b) Đếm số lần xuất hiện chuỗi con

- Hàm **y.count(x)** đếm số lần xuất hiện không giao nhau của x trong y.
- Có thể nêu các tham số xác định cụ thể phạm vi tìm kiếm. Ví dụ:

- + `y.count(x, 3)` cho biết số lần xuất hiện các xâu x không giao nhau trong xâu y nhưng chỉ trong phạm vi từ kí tự thứ ba đến kí tự cuối của xâu y.
- + `y.count(x, 3, 5)` cho biết số lần xuất hiện các xâu x không giao nhau trong xâu y nhưng chỉ trong phạm vi từ kí tự thứ ba đến kí tự thứ năm của xâu y.

c) Xác định xâu con

- Xác định xâu con của xâu **y** từ vị trí **m** đến trước vị trí **n** ($m < n$), có cú pháp:
y[m:n]

Các trường hợp đặc biệt:

- `y[:m]` là xâu con gồm **m** kí tự đầu tiên của xâu **y**.
- `y[m:]` là xâu con nhận được bằng cách bỏ **m** kí tự đầu tiên của xâu **y**.

d) Tìm vị trí xuất hiện lần đầu tiên của một xâu trong xâu khác

- Hàm **y.find(x)** trả về số nguyên xác định vị trí đầu tiên trong xâu y, từ đó xâu y xuất hiện như xâu con của y. Nếu xâu x không xuất hiện như xâu con kết quả trả về là -1.

e) Thay thế xâu con

- Hàm **y.replace(x1, x2)** tạo xâu mới từ xâu y bằng cách thay thế xâu con **x1** của y bằng xâu **x2**. Tất cả xâu con bằng **x1** và không giao nhau của y đều được thay bằng xâu **x2**.

BÀI 14: KIỂU DỮ LIỆU DANH SÁCH - XỬ LÝ DANH SÁCH

1. Kiểu dữ liệu danh sách

- Trong Python có kiểu dữ liệu danh sách (list) để lưu trữ dãy các đại lượng, ở các kiểu dữ liệu khác nhau và cho phép truy cập đến mỗi phần tử của dãy.
- Các phần tử trong danh sách của Python được đánh chỉ số bắt đầu từ 0.

Khởi tạo danh sách

Có nhiều cách khởi tạo danh sách, ba cách trong các cách đó là:

- Dùng phép gán:

Ví dụ: `ds = [1, 1, 2, 3, 5, 8]`

- Dùng câu lệnh lặp for gán giá trị trong khoảng cho trước:

Ví dụ: `ds = [i for i in range(6)]`

Kết quả: `ds = [0, 1, 2, 3, 4, 5]`

- Khởi tạo danh sách số nguyên hay thực từ dữ liệu nhập vào:

`ds = [int(i) for i in input().split()]`

`ds = [float(i) for i in`

`input().split()]` Mở rộng:

- Danh sách rỗng
- Tạo danh sách có n chữ số nguyên là 0

Truy cập đến phần tử trong danh sách

- Để chỉ định phần tử trong danh sách cần nêu tên danh sách và chỉ số phần tử đó,

chỉ số cần đặt trong dấu ngoặc vuông. Chỉ số có thể là một biểu thức số học.

2. Một số hàm và thao tác xử lý danh sách

Bảng 1. Một số hàm xử lý danh sách trong Python

Hàm xử lý danh sách	Ý nghĩa
a.append(x)	Bổ sung phần tử x vào cuối danh sách a.
a.pop(i)	Xóa phần tử đứng ở vị trí i trong danh sách a và đưa ra phần tử này.
a.insert(i,x)	Bổ sung phần tử x vào trước phần tử đứng vị trí i trong danh sách a. a.insert(0,x) sẽ bổ sung x vào đầu danh sách.
a.sort()	Sắp xếp các phần tử của danh sách a theo thứ tự không giảm.

Ghép các danh sách thành một danh sách

- Phép “+” được dùng để ghép nối hai danh sách.

Duyệt các phần tử trong danh sách theo thứ tự lưu trữ

- Gọi a là một danh sách, câu lệnh duyệt danh sách có dạng:

for i in a:

 Các câu lệnh xử lý

BÀI 16 KIỂM THỬ VÀ GỠ LỖI CHƯƠNG TRÌNH

1. Lỗi trong chương trình và kiểm thử

Hoạt động 1:

Chương trình chạy có thể có lỗi.

Kết luận:

Người lập trình thường gặp các loại lỗi như sau:

- *Lỗi cú pháp* là lỗi câu lệnh viết không theo đúng quy định của ngôn ngữ.

Ví dụ: Thiếu hoặc thừa dấu ngoặc, tên biến sai.

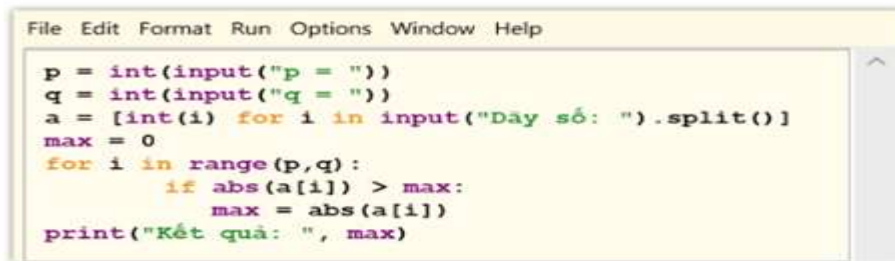
- *Lỗi ngoại lệ* (Exceptions Error, còn gọi là Runtime) là lỗi xảy ra khi chương trình đang chạy, một lệnh nào đó không thể thực hiện được.

Lỗi này không được thông báo ngay trên màn hình.

- *Lỗi ngữ nghĩa* (còn gọi là lỗi logic) là lỗi mặc dù các câu lệnh viết đúng quy định của ngôn ngữ nhưng sai trong thao tác xử lý nào đó.

Ví dụ: Nhầm tên biến, gọi hàm có tham số không đúng kiểu, thiếu câu lệnh,...

Ví dụ:



```
File Edit Format Run Options Window Help
p = int(input("p = "))
q = int(input("q = "))
a = [int(i) for i in input("Dãy số: ").split()]
max = 0
for i in range(p,q):
    if abs(a[i]) > max:
        max = abs(a[i])
print("Kết quả: ", max)
```

Hình 1a. Chương trình cần kiểm thử và tìm lỗi

Kết quả sai:



```
File Edit Shell Debug Options Window Help
p = 2
q = 4
Dãy số: 8 7 1 5 -6 4
Kết quả: 5
>>>
```

Kết quả đúng:



```
File Edit Shell Debug Options Window Help
p = 1
q = 4
Dãy số: 8 7 1 5 -6 4
Kết quả: 7
>>>
```

Để kiểm tra tính đúng đắn của chương trình so với yêu cầu đề bài, cần chuẩn bị các bộ dữ liệu đầu vào. Dữ liệu kiểm thử phải phù hợp với các ràng buộc đã cho và chia thành 3 nhóm:

- + Kiểm thử những trường hợp thường gặp trong thực tế.
- + Kiểm thử những trường hợp đặc biệt.
- + Kiểm thử những trường hợp các tham số nhận giá trị lớn nhất có thể.

2. Truy vết với cách bổ sung câu lệnh theo dõi kết quả trung gian

Hoạt động 2:

Khi đọc kỹ lại chương trình ta không biết được là thực tế máy có thực hiện đúng ý muốn của người lập trình hay không. Có thể ở một bước nào đó kết quả không như mong muốn vì chủ quan mà ta không nhận thấy được.

Kết luận:

- Một cách tìm lỗi ngữ nghĩa là bổ sung vào chương trình những câu lệnh đưa ra các kết quả trung gian nhằm truy vết các xử lý của chương trình.

3. Truy vết với công cụ gỡ lỗi của ngôn ngữ lập trình

- Trên cửa sổ Debug Control có phần hiển thị thông tin về giá trị các biến trong chương trình.

- Các thao tác để kích hoạt chế độ gỡ lỗi (SGK -tr113)

+ Trên cửa sổ **Shell**, mở file chương trình cần gỡ lỗi.

+ Chọn **Debug** chọn **Debugger**. Cửa sổ **Debug Control** xuất hiện.

+ Chọn **Run Module** (hoặc **F5**) để bắt đầu chạy chương trình.

+ Chọn **Step** để quan sát sự thay đổi các biến. Từ đó tìm lỗi sai.