

# Làm quen với Python

Vũ Đức Lý

November 19, 2016

Start

- 1 Tại sao dùng Python ? .
- 2 Thiết lập môi trường.
- 3 Cú pháp căn bản.
- 4 Phép toán căn bản.
- 5 Các cấu trúc dữ liệu trong python.
- 6 So sánh luận lý và rẽ nhánh.
- 7 Cấu trúc lặp.
- 8 Hàm trong python.
- 9 Cài đặt và sử dụng các gói thư viện trong python.

# Tại sao dùng Python ?

Python thì

- Dễ học, cú pháp đơn giản.
- Dễ dàng mở rộng.
- Có rất nhiều thư viện được hỗ trợ bởi cộng đồng .





Back

Next

Code trong các tutorial chạy python 2.7 đi kèm với hệ độ hành Ubuntu 16.04 . Trình soạn thảo Sublime Text

[Back](#)[Next](#)

# Cú pháp căn bản

- Khai báo biến 
- Kiểu giá trị boolean 
- Gán lại giá trị của biến 
- Chú thích cho code 

Back

Next

Khi tạo các ứng dụng ta làm việc với các kiểu dữ liệu khác nhau. Ví dụ, ta cần lưu một số lượng các files để quét virus:

## Example

```
number_files = 100
```

Back

# Kiểu biến boolean

Là biến mà chỉ có hai giá trị *True* (1) hoặc *False* (0). Các biến kiểu này thường được dùng để điều khiển vòng lặp hoặc được dùng như các giá trị trả về của một hàm. Nếu hàm thực hiện thành công trả về *True*, ngược lại trả về *False*

## Example

(1 equals 1)? True or False

Back

# Gán lại giá trị của biến

Đôi khi ta cần gán lại giá trị của một biến đã tồn tại một giá trị trước đó tùy theo một số điều kiện. Ví dụ, thay đổi số lượng files cần quét virus trong ví dụ trước

## Example

```
number_files = 50
```

[Back](#)



# Chú thích cho code

Khi viết code nên sử dụng các chú thích (comments) để nói mục đích của dòng code đó, điều này làm cho chương trình viết ra trở nên dễ hiểu cho mình và người đọc code.




Có 2 loại chú thích:

- Chú thích chỉ trong một dòng: bắt đầu bằng dấu #
- Chú thích trải dài trên nhiều dòng: các đoạn chú thích sẽ nằm giữa 2 dấu nháy '''

## Example

- Chú thích một dòng: # Tính tổng 2 số
- Chú thích nhiều dòng:  
''' Tính tổng 2 số  
Input: 2 số nguyên  
Output: Tổng 2 số '''

# Phép toán căn bản

- Cộng, trừ, nhân 
- Phép chia và chia lấy dư 
- Lấy số mũ 

Back

Next

# Phép cộng, trừ, nhân, chia

Các phép toán cộng, trừ, nhân trong python được biểu diễn như sau.

- Phép cộng được biểu diễn bằng dấu  $+$

## Example

```
1 + 1
```

- Phép trừ được biểu diễn bằng dấu  $-$

## Example

```
2 - 1
```

- Phép nhân được biểu diễn bằng dấu  $*$

## Example

```
2 * 1
```

[Back](#)

# Phép chia và chia có dư (modulo)

Các phép toán chia và chia lấy dư được biểu diễn trong python như sau.

- Phép chia được biểu diễn bằng dấu `\`

## Example

```
2\1
```

- Lấy số dư của phép chia dùng dấu `%`

## Example

```
3%2
```

Back

# Phép lấy số mũ





Để lấy số mũ sử dụng `**`

## Example

```
3 ** 3
```

[Back](#)

# Các cấu trúc dữ liệu trong python

- Chuỗi 
- Chỉ số chuỗi và các phép toán trên chuỗi 
- Kiểu danh sách (List) 
- Các phép toán trên danh sách 

Back

Next

# Kiểu dữ liệu chuỗi

Một chuỗi có thể chứa:

- Các kí tự, các kí tự cần được đặt trong dấu nháy kép ""

## Example

```
name = "mickey"
```

- hoặc các số, các số cần được đặt trong dấu nháy kép ""

## Example

```
age = "80"
```

- ☞ Nếu một số mà không đặt trong dấu ngoặc kép thì nó mang một giá trị thực

## Example

```
80 thì không bằng với "80"
```

# Chỉ số chuỗi và các phép toán trên chuỗi

Một chuỗi được biểu diễn bằng một mảng các kí tự:

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| m | i | c | k | e | y |
| 0 | 1 | 2 | 3 | 4 | 5 |

Trên một chuỗi (ví dụ: *name* = "mickey") cho trước ta có thể:

- Lấy ra một kí tự trong chuỗi sử dụng chỉ số: *chuỗi[chỉ số]*

## Example

```
first_char = name[0]
```

- Cộng với một chuỗi khác sử dụng dấu +

## Example

```
"mickey" + "is" + "a" + "mouse"
```

👉 python có sẵn các phương thức để thao tác trên string, ví dụ `upper()`, `lower()`, để chuyển chuỗi thành in hoa hay thường.



# Kiểu danh sách

- Kiểu danh sách có thể được sử dụng để lưu một danh sách thông tin có thứ tự. Ví dụ, một danh sách các nhân viên trong công ty, danh sách tên các máy tính có trong công ty.

|        |        |     |       |
|--------|--------|-----|-------|
| mickey | donald | tom | jerry |
| 0      | 1      | 2   | 3     |

- Trong python, một list bao gồm các phần tử được đặt trong dấu ngoặc vuông và ngăn cách nhau bởi dấu ,

## Example

```
name_list = ['mickey', 'donald', 'tom', 'jerry']
```

- Chỉ số trong danh sách giống với trường hợp chuỗi, bắt đầu từ 0.
- Một danh sách có thể bao gồm các giá trị chuỗi kí tự hoặc số.

Back

# Các phép toán trên danh sách

Trên danh sách (ví dụ: *name\_list*) cho trước ta có thể:

- Lấy ra một phần tử trong chuỗi sử dụng chỉ số: *danh sách[chỉ số]*

## Example

```
first_name = name_list[0]
```

- Xác định kích thước của danh sách sử dụng: *len(name\_list)*

## Example



```
len(name_list)
```

- Thay thế một phần tử trong danh sách dựa trên chỉ số: *danh sách [chỉ số] = giá trị mới len(name\_list)*

## Example

```
name_list[0] = "scuby"
```

# So sánh luận lý và rẽ nhánh

- Các phép so sánh luận lý 
- Rẽ nhánh 

Back

Next

## Các phép toán so sánh trong python

- So sánh bằng `==`
- So sánh không bằng `!=`
- So sánh lớn hơn `>`
- So sánh lớn hơn hoặc bằng `>=`
- So sánh nhỏ hơn `<`
- So sánh nhỏ hơn hoặc bằng `<=`

👉 Các phép so sánh sẽ trả về các giá trị luận lý (boolean) để làm cơ sở cho việc thực hiện rẽ nhánh của chương trình.

Back

Dựa trên các kết quả từ các phép so sánh để rẽ nhánh chương trình sử dụng cấu trúc *if/else*

## Example



```
if (4 > 3):  
    (căn lề) print("4 lon hon 3")  
else:  
    (căn lề) print("4 nho hon 3")
```

👉 Chú ý khi viết các phép rẽ nhánh các dòng lệnh sau dấu : cần phải được thụt vào thường là 4 khoảng trắng.

[Back](#)

# Cấu trúc lặp.

Sử dụng các cấu trúc lặp để thao tác trên các cấu trúc dữ liệu có nhiều phần tử. Ví dụ, liệt kê các file trên một thư mục, các events trong log file.

- Vòng lặp for 
- Vòng lặp while 

[Back](#)[Next](#)

# Vòng lặp for.

Sử dụng for để lặp qua các danh sách hay chuỗi. Ví dụ lặp qua danh sách các files trong thư mục, hay các events trong log file

## Example

**for** *biến lặp* **in** *đối tượng lặp*:  
(căn lề) hành động

Back

# Vòng lặp While.

Lặp lại thực hiện các hành động cho đến miễn khi điều kiện còn đúng.  
Vòng lặp while được dùng để thực hiện các lặp vô hạn (while(True))

## Example



**while** (biểu thức điều kiện):  
(căn lề) hành động

Back



# Hàm trong python.

Sử dụng hàm (function) để thực hiện nhiều đoạn mã mà lặp đi lặp lại. Việc sử dụng hàm còn giúp cho chương trình trở nên dễ hiểu về dễ dàng mở rộng.

- Định nghĩa một hàm 
- Sử dụng một hàm 

[Back](#)[Next](#)

# Định nghĩa hàm.

Định nghĩa một hàm sử dụng từ khóa *def* theo sau bởi các tham số, thân hàm bao gồm các đoạn mã cần lập và giá trị trả về (nếu có)

## Example

```
def tên_hàm(tham_số ):
    (căn_lê)các đoạn mã xử lý
    (căn_lê) return giá trị trả về
```

Back

# Gọi hàm.

Gọi hàm bằng cách gọi tên hàm và cung cấp các tham số (nếu có)

## Example

tên hàm (tham số)

Nếu hàm có giá trị trả về thì gán giá trị trả về thông qua một biến trung gian:

## Example

giá trị trả về = tên hàm (tham số)

Back

# Hàm trong python.

Python có rất nhiều các thư viện giúp thao tác với hệ thống. Để sử dụng một thư viện sử dụng cấu trúc *importtntnhtvin*

## Example

```
import os
```

Nếu thư viện chưa sẵn có trên hệ thống có thể tải về bằng nhiều cách. Dưới đây sử dụng **pip** để cài đặt

## Example

```
pip install tên thư viện
```