Number trong Python

Các đối tượng Number được tạo khi bạn gán một giá trị cho chúng.

Bạn cũng có thể xóa tham chiếu tới một đối tượng Number bởi sử dụng lệnh **del**.

del var

del var\_a, var\_b

Python hỗ trợ 4 kiểu dữ liệu số, đó là:

* **Kiểu int**: kiểu số nguyên không có dấu thập phân.
* **Kiểu long**: là các số nguyên không giới hạn kích cỡ, được theo sau bởi một chữ l hoặc chữ L.
* **Kiểu float**: số thực với dấu thập phân. Kiểu này cũng có thể được viết ở dạng số mũ của 10 với E hoặc e như (2.5e2 = 2.5 x 102 = 250).
* **Kiểu số phức**: là trong dạng a + bJ, với a và b là số thực và J (hoặc j) biểu diễn căn bậc hai của -1. Phần thực là a và phần ảo là b. Nói chung, số phức không được sử dụng nhiều trong lập trình Python.

Ngoài ra, bạn cũng cần chú ý rằng: Python cho phép bạn sử dụng chữ l với long, nhưng chúng tôi đề nghị bạn nên sử dụng chữ hoa L để tránh rắc rối với số 1.

## **Chuyển đổi kiểu số trong Python**

* Để chuyển đổi số x thành số thuần nguyên, bạn gõ **int(x)**.
* Để chuyển đổi số x thành số long, bạn gõ **long(x)**.
* Để chuyển đổi số x thành số thực, bạn gõ **float(x)**.
* Để chuyển đổi số x thành số phức với phần thực là x và phần ảo là 0, bạn gõ **complex(x)**.
* Để chuyển đổi số x và y thành số phức với phần thực là x và phần ảo là y, bạn gõ **complex(x, y)**.

## **Hằng toán học trong Python**

Python cũng định nghĩa hai hằng toán học là: hằng toán học pi và hằng toán học e.

## **Nhóm hàm toán học trong Python**

|  |  |
| --- | --- |
| **Hàm** | **Miêu tả** |
| [**Hàm abs(x)**](http://vietjack.com/python/number_abs_trong_python.jsp) | Trị tuyệt đối của x |
| [**Hàm ceil(x)**](http://vietjack.com/python/number_ceil_trong_python.jsp) | Số nguyên nhỏ nhất mà không nhỏ hơn x |
| [**Hàm cmp(x, y)**](http://vietjack.com/python/number_cmp_trong_python.jsp) | Trả về -1 nếu x < y, trả về 0 nếu x == y, hoặc 1 nếu x > y |
| [**Hàm exp(x)**](http://vietjack.com/python/number_exp_trong_python.jsp) | Trả về ex |
| [**Hàm fabs(x)**](http://vietjack.com/python/number_fabs_trong_python.jsp) | Giá trị tuyệt đối của x |
| [**Hàm floor(x)**](http://vietjack.com/python/number_floor_trong_python.jsp) | Số nguyên lớn nhất mà không lớn hơn x |
| [**Hàm log(x)**](http://vietjack.com/python/number_log_trong_python.jsp) | Trả về lnx, với x> 0 |
| [**Hàm log10(x)**](http://vietjack.com/python/number_log10_trong_python.jsp) | Trả về log10(x), với x> 0 . |
| [**Hàm max(x1, x2,...)**](http://vietjack.com/python/number_max_trong_python.jsp) | Trả về số lớn nhất |
| [**Hàm min(x1, x2,...)**](http://vietjack.com/python/number_min_trong_python.jsp) | Trả về số nhỏ nhất |
| [**Hàm modf(x)**](http://vietjack.com/python/number_modf_trong_python.jsp) | Trả về phần nguyên và phần thập phân của x. Cả hai phần có cùng dấu với x và phần nguyên được trả về dưới dạng một số thực |
| [**Hàm pow(x, y)**](http://vietjack.com/python/number_pow_trong_python.jsp) | Trả về giá trị của x\*\*y. |
| [**Hàm round(x [,n])**](http://vietjack.com/python/number_round_trong_python.jsp) | Làm tròn x về n chữ số sau dấu thập phân. Python làm tròn theo cách sau: round(0.5) là 1.0 và round(-0.5) là -1.0 |
| [**Hàm sqrt(x)**](http://vietjack.com/python/number_sqrt_trong_python.jsp) | Trả về căn bậc hai của x, với x > 0 |

## **Nhóm hàm xử lý số ngẫu nhiên trong Python**

|  |  |
| --- | --- |
| **Hàm** | **Miêu tả** |
| [**Hàm choice(seq)**](http://vietjack.com/python/number_choice_trong_python.jsp) | Một item ngẫu nhiên trong một list, tuple, hoặc một string |
| [**Hàm randrange ([start,] stop [,step])**](http://vietjack.com/python/number_randrange_trong_python.jsp) | Một phần tử được lựa chọn một cách ngẫu nhiên từ dãy (start, stop, step) |
| [**Hàm random()**](http://vietjack.com/python/number_random_trong_python.jsp) | Một số thực ngẫu nhiên r trong dãy 0>= r >1 |
| [**Hàm seed([x])**](http://vietjack.com/python/number_seed_trong_python.jsp) | Thiết lập giá trị nguyên bắt đầu mà được sử đụng trong bộ sinh số ngẫu nhiên. Bạn nên gọi hàm này trước khi gọi bất cứ hàm ngẫu nhiên nào khác. Hàm này trả về None |
| [**Hàm shuffle(lst)**](http://vietjack.com/python/number_shuffle_trong_python.jsp) | Sắp xếp các item trong list một cách ngẫu nhiên |
| [**Hàm uniform(x, y)**](http://vietjack.com/python/number_uniform_trong_python.jsp) | Một số thực ngẫu nhiên r trong dãy x>= r >y |

## **Nhóm hàm lượng giác trong Python**

|  |  |
| --- | --- |
| **Hàm** | **Miêu tả** |
| [**Hàm acos(x)**](http://vietjack.com/python/number_acos_trong_python.jsp) | Trả về arcos của x, giá trị radian |
| [**Hàm asin(x)**](http://vietjack.com/python/number_asin_trong_python.jsp) | Trả về arcsin của x, giá trị radian |
| [**Hàm atan(x)**](http://vietjack.com/python/number_atan_trong_python.jsp) | Trả về arctan của x, giá trị radian |
| [**Hàm atan2(y, x)**](http://vietjack.com/python/number_atan2_trong_python.jsp) | Trả atan(y / x), giá trị radian |
| [**Hàm cos(x)**](http://vietjack.com/python/number_cos_trong_python.jsp) | Trả về cos của x |
| [**Hàm hypot(x, y)**](http://vietjack.com/python/number_hypot_trong_python.jsp) | Trả về sqrt(x\*x + y\*y) |
| [**Hàm sin(x)**](http://vietjack.com/python/number_sin_trong_python.jsp) | Trả về sin của x |
| [**Hàm tan(x)**](http://vietjack.com/python/number_tan_trong_python.jsp) | Trả về tan của x |
| [**Hàm degrees(x)**](http://vietjack.com/python/number_degrees_trong_python.jsp) | Chuyển đổi góc x từ radian thành độ |
| [**Hàm radians(x)**](http://vietjack.com/python/number_radians_trong_python.jsp) | Chuyển đổi góc x từ độ thành radian |

# Chuỗi (String) trong Python

## **Truy cập các giá trị trong String**

* Chỉ mục với hướng forward bắt đầu với 0,1,2,3,…
* Chỉ mục với hướng backward bắt đầu với -1,-2,-3,…

Để truy cập các giá trị trong String, bạn sử dụng các dấu ngoặc vuông có chỉ mục ở bên trong. Ví dụ:

var1 = 'Hello World!'

var2 = "Python Programming"

print "var1[0]: ", var1[0]

print "var2[1:5]: ", var2[1:5]

Khi code trên được thực thi sẽ cho kết quả:

var1[0]: H

var2[1:5]: ytho

## **Cập nhật String trong Python**

Bạn có thể cập nhật một chuỗi đang tồn tại bằng cách gán (hoặc tái gán) một biến cho string khác. Giá trị mới có thể liên quan hoặc khác hoàn toàn giá trị trước đó. Ví dụ:

var1 = 'Hello World!'

print "Chuoi hien tai la :- ", var1[:6] + 'Python'

Khi code trên được thực thi sẽ cho kết quả:

Chuoi hien tai la :- Hello Python

## **Các ký tự thoát trong Python**

|  |  |  |
| --- | --- | --- |
| **Ký tự thoát** | **Biểu diễn trong hệ 16** | **Miêu tả** |
| \a | 0x07 | Bell hoặc alert |
| \b | 0x08 | Backspace |
| \cx |  | Control-x |
| \C-x |  | Control-x |
| \e | 0x1b | Escape |
| \f | 0x0c | Formfeed |
| \M-\C-x |  | Meta-Control-x |
| \n | 0x0a | Newline |
| \nnn |  | Notation trong hệ cơ số 8, ở đây n là trong dãy từ 0 tới 7 |
| \r | 0x0d | Carriage return |
| \s | 0x20 | Space |
| \t | 0x09 | Tab |
| \v | 0x0b | Tab dọc |
| \x |  | Ký tự x |
| \xnn |  | Notation trong hệ thập lục phân, ở đây n là trong dãy từ 0.9, a.f, hoặc A.F |

## **Các toán tử để thao tác với String trong Python**

Có ba kiểu toán tử được hỗ trợ bởi String, đó là:

* Toán tử cơ bản
* Toán tử membership
* Toán tử quan hệ

### **Các toán tử cơ bản để thao tác với String**

Có hai loại toán tử cơ bản có thể được sử dụng với String, đó là toán tử nối chuỗi + và toán tử lặp chuỗi \*.

**Toán tử nối chuỗi +**được sử dụng để nối hai chuỗi với nhau và tạo nên một chuỗi mới. Ví dụ:

>>> "hoang" + "nam"

Sẽ cho kết quả là:

'hoangnam'

>>>

**Chú ý**: Cả hai toán hạng được truyền cho phép nối chuỗi này phải cùng kiểu, nếu không sẽ tạo một lỗi. Ví dụ:

'abc' + 3

>>>

Sẽ tạo ra một lỗi là:

Traceback (most recent call last):

File "<pyshell#5>", line 1, in <module>

'abc' + 3

TypeError: cannot concatenate 'str' and 'int' objects

>>>

**Toán tử lặp chuỗi \*** sử dụng hai tham số. Một tham số là giá trị nguyên và tham số khác là chuỗi. Toán tử lặp chuỗi này được sử dụng để lặp đi lặp lại một chuỗi một số lần nào đó. Ví dụ:

>>> 5\*"Hoang"

Sẽ cho kết quả là:

'HoangHoangHoangHoangHoang'

**Ghi chú**: Bạn có thể sử dụng toán tử lặp chuỗi \* này theo bất kỳ cách nào như int \* string hoặc string \* int. Cả hai tham số được truyền cho toán tử này phải không trong cùng một kiểu.

### **Các toán tử membership để thao tác với String**

**Toán tử in**: trả về true nếu một ký tự là có mặt trong chuỗi đã cho, nếu không nó trả về false.

**Toán tử not in**: trả về true nếu một ký tự là không tồn tại trong chuỗi đã cho, nếu không nó trả về false.

Ví dụ:

>>> str1="javapoint"

>>> str2='sssit'

>>> str3="seomount"

>>> str4='java'

>>> st5="it"

>>> str6="seo"

>>> str4 in str1

True

>>> str5 in str2

>>> st5 in str2

True

>>> str6 in str3

True

>>> str4 not in str1

False

>>> str1 not in str4

True

### **Các toán tử quan hệ để thao tác với String**

Tất cả các toán tử quan hệ (như <,>, <=, >=, ==, !=, <>) cũng có thể áp dụng cho các String. Các chuỗi được so sánh dựa trên giá trị ASCII hoặc Unicode. Ví dụ:

>>> "HOANG"=="HOANG"

True

>>> "afsha">='Afsha'

True

>>> "Z"<>"z"

True

**Giải thích**: Giá trị ASCII của a là 97, b là 98 và c là 99, … Giá trị ASCII của A là 65, B là 66, của C là 67, … Sự so sánh giữa các chuỗi được thực hiện dựa trên giá trị ASCII.

### **Dấu chia chuỗi [] trong Python**

Có nhiều cách để chia một chuỗi. Khi chuỗi có thể được truy cập hoặc được lập chỉ mục từ cả hai hướng forward và backward thì chuỗi cũng có thể được chia theo hai hướng này. Dưới đây là cú pháp của dấu chia chuỗi [] trong Python:

<ten\_chuoi>[chi\_muc\_bat\_dau:chi\_muc\_ket\_thuc]

hoac

<ten\_chuoi>[:chi\_muc\_ket\_thuc]

hoac

<ten\_chuoi>[chi\_muc\_bat\_dau:]

Chẳng hạn với cú pháp <ten\_chuoi>[chi\_muc\_bat\_dau:chi\_muc\_ket\_thuc], thì toán tử này sẽ trả về các ký tự nằm trong dãy chỉ mục đã cho.

Ví dụ:

>>> str="Nikhil"

>>> str[0:6]

'Nikhil'

>>> str[0:3]

'Nik'

>>> str[2:5]

'khi'

>>> str[:6]

'Nikhil'

>>> str[3:]

'hil'

Ghi chú: chi\_muc\_bat\_dau trong String là inclusive, tức là bao gồm cả ký tự tại vị trí chỉ mục đó. Còn chi\_muc\_ket\_thuc là exclusive, tức là không bao gồm ký tự tại chỉ mục đó.

## **Toán tử định dạng chuỗi trong Python**

Một trong những đặc điểm hay nhất trong Python là toán tử định dạng chuỗi %. Toán tử này là duy nhất cho các String và được sử dụng với hàm print(). Ví dụ:

print "Ten toi la %s va toi nang %d kg!" % ('Hoang', 71)

Khi code trên được thực thi sẽ cho kết quả:

Ten toi la Hoang va toi nang 71 kg!

Bảng dưới đây liệt kê danh sách đầy đủ các biểu tượng có thể được sử dụng với toán tử %:

|  |  |
| --- | --- |
| **Biểu tượng định dạng** | **Chuyển đổi** |
| %c | Ký tự |
| %s | Chuyển đổi thành chuỗi thông qua hàm str() trước khi định dạng |
| %i | Số nguyên thập phân có dấu |
| %d | Số nguyên thập phân có dấu |
| %u | Số nguyên thập phân không dấu |
| %o | Số nguyên hệ bát phân |
| %x | Số nguyên hệ thập lục phân (các chữ cái thường) |
| %X | Số nguyên hệ thập lục phân (các chữ cái hoa) |
| %e | Ký hiệu số mũ (với chữ thường 'e') |
| %E | Ký hiệu số mũ (với chữ hoa 'E') |
| %f | Số thực dấu chấm động |
| %g | Viết gọn của %f và %e |
| %G | Viết gọn của %f và %E |

## **Trích dẫn tam (triple quote) trong Python**

Trích dẫn tam trong Python cho phép các chuỗi có thể trải rộng trên nhiều dòng, bao gồm đúng nguyên văn của các newline (dòng mới), tab và bất kỳ ký tự đặc biệt nào khác. Bạn theo dõi đoạn code sau:

para\_str = """day la mot chuoi day gom nhieu dong

va gom mot so ky tu khong in duoc chang han nhu

TAB ( \t ) chung se duoc hien thi dung nguyen van nhu the."""

print para\_str

Khi code trên được thực thi, nó cho kết quả như dưới đây. Bạn chú ý cách mỗi ký tự đặc biệt đã được chuyển đổi thành dạng được in của nó.

day la mot chuoi day gom nhieu dong

va gom mot so ky tu khong in duoc chang han nhu

TAB ( \t ) chung se duoc hien thi dung nguyen van nhu the.

Các chuỗi thô (raw string) không coi dấu \ như là một ký tự đặc biệt. Mỗi ký tự bạn đặt vào trong một chuỗi thô sẽ tồn tại giống như cách bạn đã viết nó.

Để hiểu rõ vấn đề này, trước hết bạn theo dõi ví dụ:

print 'C:\\nowhere'

Khi code trên được thực thi sẽ cho kết quả:

C:\nowhere

Bây giờ sử dụng chuỗi thô. Chúng ta đã đặt biểu thức trong r'bieu\_thuc' như sau:

print r'C:\\nowhere'

Khi code trên được thực thi sẽ cho kết quả:

C:\\nowhere

## **Chuỗi dạng Unicode trong Python**

Các chuỗi thông thường trong Python được lưu trữ nội tại dưới dạng ASCII 8 bit, trong khi các chuỗi Unicode được lưu trữ dưới dạng Unicode 16 bit. Điều này cho phép để có một tập hợp các ký tự đa dạng hơn, bao gồm các ký tự đặc biệt từ hầu hết các ngôn ngữ trên thế giới. Bạn theo dõi ví dụ:

print u'Hello, world!'

Khi code trên được thực thi sẽ cho kết quả:

Hello, world!

Như bạn có thể thấy, các chuỗi dạng Unicode sử dụng tiền tố u, trong khi các chuỗi thô sử dụng tiền tố r.

## **Các phương thức và hàm đã xây dựng sẵn để xử lý chuỗi trong Python**

Python cung cấp các phương thức đa dạng đã được xây dựng sẵn để thao tác với các chuỗi. Bảng dưới đây liệt kê các phương thức này. Bạn truy cập link để thấy ví dụ chi tiết.

|  |  |
| --- | --- |
| **STT** | **Phương thức và Miêu tả** |
| 1 | [**Phương thức capitalize()**](http://vietjack.com/python/string_capitalize_trong_python.jsp) Viết hoa chữ cái đầu tiên của chuỗi |
| 2 | [**Phương thức center(width, fillchar)**](http://vietjack.com/python/string_center_trong_python.jsp)  Trả về một chuỗi mới, trong đó chuỗi ban đầu đã được cho vào trung tâm và hai bên đó là các fillchar sao cho tổng số ký tự của chuỗi mới là width |
| 3 | [**Phương thức count(str, beg= 0,end=len(string))**](http://vietjack.com/python/string_count_trong_python.jsp)  Đếm xem chuỗi str này xuất hiện bao nhiêu lần trong chuỗi string hoặc chuỗi con của string nếu bạn cung cấp chỉ mục ban đầu start và chỉ mục kết thúc end |
| 4 | [**Phương thức decode(encoding='UTF-8',errors='strict')**](http://vietjack.com/python/string_decode_trong_python.jsp)  Giải mã chuỗi bởi sử dụng encoding đã cho |
| 5 | [**Phương thức encode(encoding='UTF-8',errors='strict')**](http://vietjack.com/python/string_encode_trong_python.jsp)  Trả về phiên bản chuỗi đã được mã hóa của chuỗi ban đầu. Nếu có lỗi xảy ra, thì chương trình sẽ tạo một ValueError trừ khi các lỗi này được cung cấp với ignore hoặc replace |
| 6 | [**Phương thức endswith(suffix, beg=0, end=len(string))**](http://vietjack.com/python/string_endswith_trong_python.jsp) Xác định xem nếu chuỗi string hoặc chuỗi con đã cho của string (nếu bạn cung cấp chỉ mục bắt đầu beg và chỉ mục kết thúc end) kết thúc với hậu tố suffix thì trả về true, nếu không thì phương thức này trả về false |
| 7 | [**Phương thức expandtabs(tabsize=8)**](http://vietjack.com/python/string_expandtabs_trong_python.jsp)  Mở rộng các tab trong chuỗi tới số khoảng trống đã cho; mặc định là 8 space cho mỗi tab nếu bạn không cung cấp tabsize |
| 8 | [**Phương thức find(str, beg=0 end=len(string))**](http://vietjack.com/python/string_find_trong_python.jsp)  Xác định xem chuỗi str có xuất hiện trong chuỗi string hoặc chuỗi con đã cho của string (nếu bạn cung cấp chỉ mục bắt đầu beg và chỉ mục kết thúc end), nếu xuất hiện thì trả về chỉ mục của str, còn không thì trả về -1 |
| 9 | [**Phương thức index(str, beg=0, end=len(string))**](http://vietjack.com/python/string_index_trong_python.jsp)  Tương tự như find(), nhưng tạo ra một ngoại lệ nếu str là không được tìm thấy |
| 10 | [**Phương thức isalnum()**](http://vietjack.com/python/string_isalnum_trong_python.jsp)  Trả về true nếu chuỗi có ít nhất một ký tự và tất cả ký tự là chữ-số. Nếu không hàm sẽ trả về false |
| 11 | [**Phương thức isalpha()**](http://vietjack.com/python/string_isalpha_trong_python.jsp)  Trả về true nếu chuỗi có ít nhất 1 ký tự và tất cả ký tự là chữ cái. Nếu không phương thức sẽ trả về false |
| 12 | [**Phương thức isdigit()**](http://vietjack.com/python/string_isdigit_trong_python.jsp)  Trả về true nếu chuỗi chỉ chứa các chữ số, nếu không là false |
| 13 | [**Phương thức islower()**](http://vietjack.com/python/string_islower_trong_python.jsp)  Trả về true nếu tất cả ký tự trong chuỗi là ở dạng chữ thường, nếu không là false |
| 14 | [**Phương thức isnumeric()**](http://vietjack.com/python/string_isnumeric_trong_python.jsp)  Trả về true nếu một chuỗi dạng Unicode chỉ chứa các ký tự số, nếu không là false |
| 15 | [**Phương thức isspace()**](http://vietjack.com/python/string_isspace_trong_python.jsp)  Trả về true nếu chuỗi chỉ chứa các ký tự khoảng trắng whitespace, nếu không là false |
| 16 | [**Phương thức istitle()**](http://vietjack.com/python/string_istitle_trong_python.jsp)  Trả về true nếu chuỗi là ở dạng titlecase, nếu không là false |
| 17 | [**Phương thức isupper()**](http://vietjack.com/python/string_isupper_trong_python.jsp)  Trả về true nếu tất cả ký tự trong chuỗi là chữ hoa |
| 18 | [**Phương thức join(seq)**](http://vietjack.com/python/string_join_trong_python.jsp)  Nối chuỗi các biểu diễn chuỗi của các phần tử trong dãy seq thành một chuỗi |
| 19 | [**Phương thức len(string)**](http://vietjack.com/python/string_len_trong_python.jsp)  Trả về độ dài của chuỗi |
| 20 | [**Phương thức ljust(width[, fillchar])**](http://vietjack.com/python/string_ljust_trong_python.jsp)  Trả về một chuỗi mới, trong đó có chuỗi ban đầu được căn chỉnh vào bên trái và bên phải là các fillchar sao cho tổng số ký tự là width |
| 21 | [**Phương thức lower()**](http://vietjack.com/python/string_lower_trong_python.jsp)  Chuyển đối tất cả chữ hoa trong chuỗi sang kiểu chữ thường |
| 22 | [**Phương thức lstrip()**](http://vietjack.com/python/string_lstrip_trong_python.jsp)  Xóa tất cả các khoảng trống trắng ban đầu (leading) trong chuỗi |
| 23 | [**Phương thức maketrans()**](http://vietjack.com/python/string_maketrans_trong_python.jsp)  Trả về một bảng thông dịch được sử dụng trong hàm translate |
| 24 | [**Phương thức max(str)**](http://vietjack.com/python/string_max_trong_python.jsp)  Trả về ký tự chữ cái lớn nhất từ chuỗi str đã cho |
| 25 | [**Phương thức min(str)**](http://vietjack.com/python/string_min_trong_python.jsp)  Trả về ký tự chữ cái nhỏ nhất từ chuỗi str đã cho |
| 26 | [**Phương thức replace(old, new [, max])**](http://vietjack.com/python/string_replace_trong_python.jsp)  Thay thế tất cả sự xuất hiện của old trong chuỗi với new với số lần xuất hiện max (nếu cung cấp) |
| 27 | [**Phương thức rfind(str, beg=0,end=len(string))**](http://vietjack.com/python/string_rfind_trong_python.jsp)  Tương tự hàm find(), nhưng trả về chỉ mục cuối cùng |
| 28 | [**Phương thức rindex( str, beg=0, end=len(string))**](http://vietjack.com/python/string_rindex_trong_python.jsp)  Giống index(), nhưng trả về chỉ mục cuối cùng nếu tìm thấy |
| 29 | [**Phương thức rjust(width,[, fillchar])**](http://vietjack.com/python/string_rjust_trong_python.jsp)  Trả về một chuỗi mới, trong đó có chuỗi ban đầu được căn chỉnh vào bên phải và bên trái là các fillchar sao cho tổng số ký tự là width |
| 30 | [**Phương thức rstrip()**](http://vietjack.com/python/string_rstrip_trong_python.jsp)  Xóa bỏ tất cả các khoảng trống trắng ở cuối (trailing) của chuỗi |
| 31 | [**Phương thức split(str="", num=string.count(str))**](http://vietjack.com/python/string_split_trong_python.jsp)  Chia chuỗi theo delimeter đã cho (là space nếu không được cung cấp) và trả về danh sách các chuỗi con; nếu bạn cung cấp num thì chia chuỗi thành num chuỗi con |
| 32 | [**Phương thức splitlines( num=string.count('\n'))**](http://vietjack.com/python/string_splitlines_trong_python.jsp)  Trả về một List gồm tất cả các dòng trong chuỗi, và tùy ý xác định các ngắt dòng (nếu num được cung cấp và là true). |
| 33 | [**Phương thức startswith(str, beg=0,end=len(string))**](http://vietjack.com/python/string_startswith_trong_python.jsp)  Xác định xem chuỗi hoặc chuỗi con (nếu bạn cung cấp chỉ mục bắt đầu beg và chỉ mục kết thúc end) có bắt đầu với chuỗi con str không, nếu có trả về true, nếu không là false |
| 34 | [**Phương thức strip([chars])**](http://vietjack.com/python/string_strip_trong_python.jsp)  Thực hiện cả hai phương thức lstrip() và rstrip() trên chuỗi |
| 35 | [**Phương thức swapcase()**](http://vietjack.com/python/string_swapcase_trong_python.jsp)  Đảo ngược kiểu của tất cả ký tự trong chuỗi |
| 36 | [**Phương thức title()**](http://vietjack.com/python/string_title_trong_python.jsp)  Trả về một bản sao của chuỗi trong đó tất cả ký tự đầu tiên của tất cả các từ là ở kiểu chữ hoa. |
| 37 | [**Phương thức translate(table, deletechars="")**](http://vietjack.com/python/string_translate_trong_python.jsp)  Trả về một bản sao đã được thông dịch của chuỗi |
| 38 | [**Phương thức upper()**](http://vietjack.com/python/string_upper_trong_python.jsp)  Chuyển đổi các chữ thường trong chuỗi thành chữ hoa |
| 39 | [**Phương thức zfill (width)**](http://vietjack.com/python/string_zfill_trong_python.jsp)  Trả về một chuỗi mới, trong đó bao gồm chuỗi ban đầu và được đệm thêm với các số 0 vào bên trái sao cho tổng ký tự là width |
| 40 | [**Phương thức isdecimal()**](http://vietjack.com/python/string_isdecimal_trong_python.jsp)  Trả về true nếu một chuỗi dạng Unicode chỉ chứa các ký tự thập phân, nếu không là false |

# List trong Python

<ten\_list>=[giatri1, giatri2, ..., giatriN];

Ví dụ:

list1 = ['vatly', 'hoahoc', 1997, 2000];

list2 = [1, 2, 3, 4, 5 ];

list3 = ["a", "b", "c", "d"];

Tương tự như chỉ mục của chuỗi, chỉ mục của List bắt đầu từ 0.

## **Truy cập các giá trị trong List trong Python**

Để truy cập các giá trị trong List, bạn sử dụng cú pháp sau:

<ten\_list>[index]

để lấy giá trị có sẵn tại chỉ mục đó.

Ví dụ:

list1 = ['vatly', 'hoahoc', 1997, 2000];

list2 = [1, 2, 3, 4, 5, 6, 7 ];

print "list1[0]: ", list1[0]

print "list2[1:5]: ", list2[1:5]

Khi code trên được thực thi sẽ cho kết quả:

list1[0]: vatly

list2[1:5]: [2, 3, 4, 5]

## **Các hoạt động cơ bản trên List trong Python**

Ví dụ cho nối List:

list1=[10,20]

list2=[30,40]

list3=list1+list2

print list3

Kết quả là:

>>>

[10, 20, 30, 40]

>>>

Ví dụ cho lặp List:

list1=[10,20]

print list1\*1

Kết quả là:

>>>

[10, 20]

>>>

## **Cập nhật List trong Python**

Bạn có thể cập nhật một hoặc nhiều phần tử của List bởi gán giá trị cho chỉ mục cụ thể đó. Cú pháp:

<ten\_list>[index]=<giatri>

Ví dụ:

list = ['vatly', 'hoahoc', 1997, 2000];

print "Gia tri co san tai chi muc thu 2 : "

print list[2]

list[2] = 2001;

print "Gia tri moi tai chi muc thu 2 : "

print list[2]

Khi code trên được thực thi sẽ cho kết quả:

Gia tri co san tai chi muc thu 2 :

1997

Gia tri moi tai chi muc thu 2 :

2001

## **Phụ thêm phần tử vào cuối một List**

Phương thức append() được sử dụng để phụ thêm phần tử vào cuối một List. Cú pháp:

<ten\_list>.append(item)

Ví dụ:

list1=[10,"hoang",'z']

print "Cac phan tu cua List la: "

print list1

list1.append(10.45)

print "Cac phan tu cua List sau khi phu them la: "

print list1

Khi code trên được thực thi sẽ cho kết quả:

>>>

Cac phan tu cua List la:

[10, 'hoang', 'z']

Cac phan tu cua List sau khi phu them la:

[10, 'hoang', 'z', 10.45]

>>>

## **Xóa phần tử trong List**

Để xóa một phần tử trong List, bạn có thể sử dụng lệnh del nếu bạn biết chính xác phần tử nào bạn muốn xóa hoặc sử dụng phương thức remove() nếu bạn không biết. Ví dụ:

list1 = ['vatly', 'hoahoc', 1997, 2000];

print list1

del list1[2];

print "Cac phan tu cua List sau khi xoa gia tri tai chi muc 2 : "

print list1

Khi code trên được thực thi sẽ cho kết quả:

['vatly', 'hoahoc', 1997, 2000]

Cac phan tu cua List sau khi xoa gia tri tai chi muc 2 :

['vatly', 'hoahoc', 2000]

Bạn cũng có thể sử dụng del để xóa tất cả phần tử từ chi\_muc\_bat\_dau tới chi\_muc\_ket\_thuc như sau:

list1=[10,'hoang',50.8,'a',20,30]

print list1

del list1[0]

print list1

del list1[0:3]

print list1

Khi code trên được thực thi sẽ cho kết quả:

>>>

[10, 'hoang', 50.8, 'a', 20, 30]

['hoang', 50.8, 'a', 20, 30]

[20, 30]

>>>

## **Các hàm và phương thức đã xây dựng sẵn để xử lý List trong Python**

|  |  |
| --- | --- |
| **TT** | **Hàm và Miêu tả** |
| 1 | [**Hàm cmp(list1, list2)**](http://vietjack.com/python/list_cmp_trong_python.jsp)  So sánh các phần tử trong cả hai list |
| 2 | [**Hàm len(list)**](http://vietjack.com/python/list_len_trong_python.jsp)  Trả về độ dài của list |
| 3 | [**Hàm max(list)**](http://vietjack.com/python/list_max_trong_python.jsp)  Trả về phần tử có giá trị lớn nhất trong list |
| 4 | [**Hàm min(list)**](http://vietjack.com/python/list_min_trong_python.jsp)  Trả về phần tử có giá trị nhỏ nhất trong list |
| 5 | [**Hàm list(seq)**](http://vietjack.com/python/list_list_trong_python.jsp)  Chuyển đổi một tuple thành list |

Danh sách các phương thức xử lý List trong Python:

|  |  |
| --- | --- |
| **STT** | **Phương thức và Miêu tả** |
| 1 | [**Phương thức list.append(obj)**](http://vietjack.com/python/list_append_trong_python.jsp)  Phụ thêm đối tượng obj vào cuối list |
| 2 | [**Phương thức list.count(obj)**](http://vietjack.com/python/list_count_trong_python.jsp)  Đếm xem có bao nhiêu lần mà obj xuất hiện trong list |
| 3 | [**Phương thức list.extend(seq)**](http://vietjack.com/python/list_extend_trong_python.jsp)  Phụ thêm các nội dung của seq vào cuối list |
| 4 | [**Phương thức list.index(obj)**](http://vietjack.com/python/list_index_trong_python.jsp)  Trả về chỉ mục thấp nhất trong list mà tại đó obj xuất hiện |
| 5 | [**Phương thức list.insert(index, obj)**](http://vietjack.com/python/list_insert_trong_python.jsp)  Chèn đối tượng obj vào trong list tại index đã cho |
| 6 | [**Phương thức list.pop(obj=list[-1])**](http://vietjack.com/python/list_pop_trong_python.jsp)  Xóa và trả về phần tử cuối cùng hoặc đối tượng obj có chỉ mục đã cung cấp từ list đã cho |
| 7 | [**Phương thức list.remove(obj)**](http://vietjack.com/python/list_remove_trong_python.jsp)  Xóa đối tượng obj từ list |
| 8 | [**Phương thức list.reverse()**](http://vietjack.com/python/list_reverse_trong_python.jsp)  Đảo ngược thứ tự các đối tượng trong list |
| 9 | [**Phương thức list.sort([func])**](http://vietjack.com/python/list_sort_trong_python.jsp)  Sắp xếp các đối tượng của list, sử dụng hàm so sánh nếu được cung cấp |

# Tuple trong Python

Không giống List mà sử dụng các dấu ngoặc vuông, thì tuple sử dụng các dấu ngoặc đơn. Các đối tượng trong tuple được phân biệt bởi dấu phảy và được bao quanh bởi dấu ngoặc đơn (). Giống như chỉ mục của chuỗi, chỉ mục của tuple bắt đầu từ 0.

Ví dụ:

>>> data=(10,20,'ram',56.8)

>>> data2="a",10,20.9

>>> data

(10, 20, 'ram', 56.8)

>>> data2

('a', 10, 20.9)

>>>

**Ghi chú**: Nếu các dấu ngoặc đơn không được cung cấp với một dãy, thì nó được coi như là tuple.

Một tuple trống không chứa phần tử nào, ví dụ:

tup1 = ();

Với một tuple chỉ có một giá trị đơn, thì phải có một dấu phảy ở cuối, ví dụ:

tup1 = (50,);

Các tuple cũng có thể được lồng vào nhau, ví dụ:

tupl1='a','hoang',10.56

tupl2=tupl1,(10,20,30)

print tupl1

print tupl2

Kết quả:

>>>

('a', 'hoang', 10.56)

(('a', 'hoang', 10.56), (10, 20, 30))

>>>