

## Danh sách bài học

## Định dạng chuỗi ký tự trong Python

Trong các bài trước, chúng ta đã làm việc với các chuỗi ký tự đơn giản. Tuy nhiên, trên thực tế có rất nhiều bài toán buộc chúng ta phải định dạng chuỗi ký tự chẳng hạn như căn trái, căn phải hoặc hiển thị ra những chuỗi ký tự có định dạng đặc biệt. Trong những trường hợp như vậy, chúng ta phải làm như thế nào? Trong bài này, chúng ta sẽ cùng tìm hiểu cách thức để định dạng các chuỗi như vậy trong Python.

### Định dạng chuỗi ký tự kiểu cũ

Python kế thừa cách định dạng chuỗi ký tự khi in ra theo cách tương tự như các ngôn ngữ lập trình cũ hơn như C hay C++. Chúng ta có thể sử dụng ký hiệu dấu phần trăm (%) để thực hiện điều này.

**Ví dụ 1:**

```
1 | a = 100.375
2 | print('%.3f' %a)
3 | print('%.5f' %a)
```



[Copy](#)

Kết quả:

```
1 | 100.375
2 | 100.37500
```

Ở đây, trong câu lệnh thứ nhất chúng ta chỉ định chuỗi in ra phải có độ chính xác đến 3 chữ số phía sau, còn ở câu lệnh thứ hai chúng ta chỉ định độ chính xác này là 5 chữ số. Dấu % thể hiện rằng đây là một chỉ định định dạng ký tự, f thể hiện rằng chúng ta định dạng chuỗi theo dạng hiển thị của số thực dấu chấm động dạng decimal. Bảng dưới đây sẽ thể hiện rõ hơn về các ký tự có thể sử dụng để định dạng và ý nghĩa của nó:

Ký tự	Ý nghĩa	Ví dụ
'd'	Số nguyên có dấu	<pre>print('%d' %(-100)) &gt;&gt;&gt; -100</pre>
'i'	Số nguyên có dấu	<pre>print('%i' %(-100)) &gt;&gt;&gt; -100</pre>
'o'	Giá trị dạng bát phân (octan)	<pre>print('%o' %(20)) &gt;&gt;&gt; 24</pre>
'u'	Giống với ký tự 'd'.	<pre>print('%u' %(-100))</pre>

Ký tự	Ý nghĩa	Ví dụ
		>>> -100
'x'	Giá trị hệ 16 có dấu (viết thường)	print("%5x"% (47)) >>> 2f
'X'	Giá trị hệ 16 có dấu (viết hoa)	print("%5.4X"% (47)) >>> 002F
'e'	Định dạng số mũ cho số thực dấu chấm động (viết thường)	print("%9.2e"% (312.087)) >>> 3.12e+02
'E'	Định dạng số mũ cho số thực dấu chấm động (viết hoa)	print("%9.2E"% (312.087)) >>> 3.12E+02
'f'	Định dạng số thực dạng thập phân	print('%f' %(100.21)) >>> 100.210000
'F'	Định dạng số thực dạng thập phân	print('%F' %(100.21)) >>> 100.210000
'g'	Định dạng số thực dấu chấm động. Sử dụng định dạng số mũ viết thường nếu số mũ nhỏ hơn -4 hoặc nhỏ hơn độ chính xác	print('%g' %(3e9)) >>> 3e+09
'G'	Định dạng số thực dấu chấm động. Sử dụng định dạng số mũ viết hoa nếu số mũ nhỏ hơn -4 hoặc nhỏ hơn độ chính xác	print('%G' %(3e9)) >>> 3E+09
'c'	Một ký tự (nhận vào một giá trị số nguyên hoặc một ký tự)	print('%c' %('a')) >>> a
'r'	Chuỗi ký tự (chuyển đổi từ bất kỳ đối tượng Python nào bằng hàm repr())	print('%r' %('Python')) >>> 'Python'
's'	Chuỗi ký tự (chuyển đổi từ bất kỳ đối tượng Python nào bằng hàm str())	print('%s' %('Python')) >>> Python
'a'	Chuỗi ký tự (chuyển đổi từ bất kỳ đối tượng Python nào bằng hàm ascii())	print('%a' %('Python cơ bản')) >>> 'Python c\u0111\u01b1c b\u01ea3n'  

Ký tự	Ý nghĩa	Ví dụ
'%'	Không có tham số nào được chuyển đổi, trả về kết quả là ký tự dấu phần trăm '%'	<pre>print('%') &gt;&gt;&gt; %</pre>

## Định dạng sử dụng phương thức format()

Để linh hoạt và dễ dàng hơn cho việc định dạng, Python 3 hỗ trợ phương thức định dạng chuỗi `format()`. Phương thức `format()` có tính rất linh hoạt và mạnh mẽ trong việc định dạng chuỗi ký tự. Chuỗi ký tự được định dạng sẽ bao gồm cặp dấu ngoặc nhọn để đặt vị trí cho giá trị sẽ được in ra. Ngoài ra, chúng ta có thể sử dụng đối số vị trí hoặc đối số từ khóa để chỉ định thứ tự cho giá trị được in ra. Phương thức `format()` có nhiệm vụ thực hiện định dạng (các) giá trị được chỉ định và chèn các giá trị này vào bên trong đối tượng giữ chỗ của chuỗi ký tự. Đối tượng giữ chỗ được xác định bằng dấu ngoặc nhọn: `{}`.

**Cú pháp của phương thức này như sau:**

```
1 | string.format(value1, value2...)
```

Trong đó:

- Tham số `value1` và `value2` là bắt buộc. Một hoặc nhiều giá trị cần được định dạng và chèn vào chuỗi ký tự. Các giá trị là danh sách các giá trị được phân tách bằng dấu phẩy. Các giá trị có thể thuộc bất kỳ kiểu dữ liệu nào.

**Ví dụ 2:**

```
1 | print("My name is {}, you can learn about me.".format("Python"))
2 | print("{} is easy with {}".format("Python", "tek4.vn"))
3 | print("{str} is easy with {str2}.".format(str = "Python", str2 = "tek4.vn"))
```

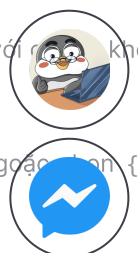
Kết quả đầu ra:

```
1 | My name is Python, you can learn about me.
2 | Python is easy with tek4.vn
3 | Python is easy with tek4.vn
```

- Trong đoạn mã bên trên, câu lệnh `print()` đầu tiên sẽ thực hiện in ra chuỗi ký tự với các đối tượng giữ chỗ tương ứng (Python).
- Câu lệnh `print()` thứ hai có sử dụng các đối tượng giữ chỗ với giá trị số nguyên được truyền vào, tương ứng với các vị trí của chuỗi ký tự trong phương thức `format()` là 2 giá trị của `{0}` và `{1}`. Các giá trị của biến, biểu thức nằm trong đối số của phương thức `format()` sẽ được lần lượt thay thế vào đây (theo đúng thứ tự).
- Câu lệnh `print()` thứ ba có sử dụng các đối tượng giữ chỗ với các từ khóa `{str}` và `{str2}`, tương ứng với từ khóa trong phương thức `format()`.

Để định dạng các chuỗi ký tự một cách mềm dẻo hơn, chúng ta thêm các chỉ thị định dạng vào trong dấu ngoặc nhọn `{}`.  
Chẳng hạn:

**Ví dụ 3:**



```

1 | print("Number {:d}".format(1982))
2 | print("Float number: {:.f}".format(395.199999))
3 | print("Binary: {0:b}, Oct: {0:o}, Hex: {0:x}".format(19))

```

Kết quả đầu ra:

```

1 | Number 1982
2 | Float number: 395.199990
3 | Binary: 10011, Oct: 23, Hex: 13

```

- Trong đoạn mã bên trên, câu lệnh `print()` đầu tiên sẽ thực hiện in ra định dạng số nguyên (với chỉ thị `:d`).
- Câu lệnh `print()` thứ hai in ra giá trị số thực dạng thập phân (chỉ thị `:f`).
- Câu lệnh `print()` cuối cùng thực hiện in ra các định dạng cơ số 2 (chỉ thị `:b`), 8 (chỉ thị `:o`) và 16 (chỉ thị `:x`) của giá trị 19.

Các chỉ thị định dạng chi tiết có thể xem trong bảng dưới đây:

Kiểu định dạng	Mô tả	Ví dụ
<code>:&lt;</code>	Căn lề bên trái với một lượng khoảng trống nhất định	<pre>print("There are {:&lt;8} chickens".format(49)) &gt;&gt;&gt; There are 49 chickens</pre>
<code>:&gt;</code>	Căn lề bên phải với một lượng khoảng trống nhất định	<pre>print("There are {:&gt;8} chickens".format(49)) &gt;&gt;&gt; There are 49 chickens</pre>
<code>:^</code>	Căn lề chính giữa với một lượng khoảng trống nhất định	<pre>print("There are {:^8} chickens".format(49)) &gt;&gt;&gt; There are 49 chickens</pre>
<code>:=</code>	Đặt dấu ở vị trí bên trái xa nhất	<pre>print("It's {::=10} degrees".format(-10)) &gt;&gt;&gt; It's - 10 degrees</pre>
<code>:+</code>	Chỉ định kết quả là số dương hay số âm	<pre>print("It is {:+} and {:+} degrees".format(-3, 7)) &gt;&gt;&gt; It is -3 and +7 degrees</pre>
<code>:-</code>	Chỉ định cho các kết quả là số âm	<pre>print("It is {: -} and {: -} degrees celsius.".format(-3, 7)) &gt;&gt;&gt; It is -3 and 7 degrees celsius.</pre>
<code>:</code>	Sử dụng một khoảng trắng để thêm khoảng trắng trước các giá trị số dương	<pre>print("It is { } and { } degrees celsius.".format(-3, 7)) &gt;&gt;&gt; It is -3 and 7 degrees celsius.</pre> 
<code>:,</code>	Sử dụng dấu phẩy làm dấu phân tách hàng nghìn	<pre>print("Universe is {:,} years old.".format(13800000000)) &gt;&gt;&gt; Universe is 13,800,000,000 years old.</pre> 

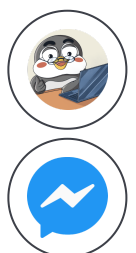
:_	Sử dụng dấu gạch dưới làm dấu phân tách hàng nghìn	<pre>print("The universe is {:.} years old.".format(13800000000)) &gt;&gt;&gt; The universe is 13_800_000_000 years old.</pre>
:b	Định dạng số nhị phân	<pre>print("Binary of {0} is {0:b}".format(5)) &gt;&gt;&gt; Binary of 5 is 101</pre>
:c	Chuyển đổi giá trị sang các bộ ký tự unicode tương ứng	<pre>print('It is {:c}'.format(64)) &gt;&gt;&gt; It is @</pre>
:d	Định dạng số thập phân	<pre>print("There are {:d} chickens.".format(0b101)) &gt;&gt;&gt; There are 5 chickens.</pre>
:e	Định dạng số học, với ký tự e viết thường	<pre>print("There are {:e} chickens.".format(5)) &gt;&gt;&gt; There are 5.000000e+00 chickens.</pre>
:f	Cố định định dạng số thập phân	<pre>print("Price is {:.2f}\$".format(45)) &gt;&gt;&gt; Price is 45.00\$</pre>
:g	Định dạng chung	<pre>print('{:g}'.format(3.14159)) &gt;&gt;&gt; 3.14159</pre>
:o	Định dạng số bát phân	<pre>print("Octal of {0} is {0:o}".format(10)) &gt;&gt;&gt; Octal of 10 is 12</pre>
:x	Định dạng số hệ 16	<pre>print("Hexadecimal of {0} is {0:x}".format(255)) &gt;&gt;&gt; Hexadecimal of 255 is ff</pre>
:n	Định dạng số	<pre>print('{:n}'.format(359)) &gt;&gt;&gt; 359</pre>
:%	Định dạng số phần trăm	<pre>print("Score is {:.0%}".format(0.25)) &gt;&gt;&gt; Score is 25%</pre>

Ngoài ra, chúng ta có thể thêm các con số để định dạng số lượng vị trí hiển thị cũng như độ chính xác của số hiển thị.

#### Ví dụ 4:

```
1 print("{:5d}".format(19))
2 print("{:2d}".format(3291))
3 print("{:8.3f}".format(19.9813219))
4 print("{:05d}".format(21))
5 print("{:08.3f}".format(10.438292189))
```

Kết quả đầu ra:



```

1 | 19
2 | 3291
3 | 19.981
4 | 00021
5 | 0010.438

```

Trong đoạn mã bên trên:

- Câu lệnh `print()` thứ nhất hiện ra số nguyên với độ rộng hiển thị là 5, nếu số ký tự thực tế nhỏ hơn độ rộng hiển thị thì trên màn hình sẽ đệm thêm các khoảng trắng để hiển thị đủ 5 vị trí.
- Với câu lệnh `print()` phía sau khi thừa vị trí hiển thị (4 so với 2) thì chương trình sẽ tự động hiển thị số lượng cần thiết chứ không cắt bớt.
- Câu lệnh `print()` thứ ba thực hiện in ra số thập phân với khoảng rộng có thể hiển thị là 8, và thực hiện làm tròn đến chữ số thứ 3 sau dấu thập phân.
- Ở dòng thứ 4 và thứ 5, thay vì đệm vị trí thiếu bằng khoảng trắng thì ta đệm bằng các số 0.

Để căn trái và căn giữa chúng ta sử dụng các ký tự `>`, `<`, `^`.

#### Ví dụ 5:

```

1 | print("-----")
2 | print("{:20}Python".format("Python"))
3 | print("{:>20}".format("Python"))
4 | print("{:^20}".format("Python"))

```

Kết quả đầu ra:

```

1 | -----
2 | Python           Python
3 |           Python
4 |       Python

```

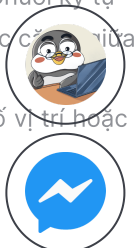
Trong đoạn mã trên:

- Câu lệnh `print()` đầu tiên sẽ thực hiện in ra chuỗi ký tự có nhiệm vụ để giống hàng cho chúng ta dễ nhìn.
- Câu lệnh `print()` thứ hai thực hiện in chuỗi ký tự được căn chỉnh bên trái và có khoảng trống là 20.
- Câu lệnh `print()` thứ 3 và 4 thực hiện in ra chuỗi ký tự được căn chỉnh bên phải và giữa với khoảng trống là 20.

Chúng ta sẽ sử dụng 20 dấu gạch ngang để đại diện cho khoảng trống cho việc minh họa. Các bạn có thể thấy rằng chuỗi ký tự "Python" đầu tiên được đặt ở bên trái, chuỗi ký tự "Python" thứ 2 được đặt sau khoảng trống thứ 20. Chuỗi ký tự "Python" thứ 3 được đặt ở bên phải và có khoảng trống còn lại bên trái là 15. Chuỗi "Python" cuối cùng được đặt ở giữa trong khoảng trống là 20.

Ngoài ra, bạn cũng có thể truyền các mã định dạng như độ chính xác, căn chỉnh, điền ký tự dưới dạng đối số vị trí hoặc từ khóa một cách linh động.

#### Ví dụ 6:



```
1 | string = "{:{fill}{align}{width}}"
2 | print(string.format('python', fill='*', align='^', width=9))
3 | num = "{:{align}{width}.{precision}f}"
4 | print(num.format(392.989, align='<', width=8, precision=1))
```

Kết quả đầu ra:

```
1 | *python**
2 | 393.0
```

Trong ví dụ đầu tiên, 'python' là đối số vị trí cần được định dạng. Tương tự như vậy, fill = '\*', align = '^' và width = 9 là các đối số từ khóa. Trong chuỗi ký tự mẫu, các đối số từ khóa này không được truy xuất như các chuỗi ký tự bình thường được in ra mà sẽ quyết định các mã định dạng fill, align và width. Các đối số thay thế các đối tượng giữ chỗ được đặt tên tương ứng và chuỗi 'python' được định dạng tương ứng.

Tương tự như vậy, trong ví dụ thứ hai, 392.989 là đối số vị trí và align, width và precision được truyền cho chuỗi ký tự mẫu dưới dạng mã định dạng.

Tiếp theo chúng ta hãy làm quen với việc định dạng chuỗi ký tự in ra với các cách thức đã học.

Hoàn thành bài học

