Bài đọc thêm: Định dạng chuỗi ký tự trong Python

Nội dung

Sử dụng thư viện String ta cần lệnh:

```
import String
```

Các hằng số có sẵn trong module String:

`string.`**`ascii_lowercase`**

Gồm các ký tự chữ cái in thường trong bảng ascii

```
ascii_lowercase = 'abcdefghijklmnopqrstuvwxyz'
```

`string.`**`ascii_uppercase`**

Gồm các ký tự chữ cái in hoa trong bảng ascii

```
ascii_uppercase = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

`string.`**`ascii_letters`**

Gồm các ký tự chữ cái có trong bảng mã ascii

```
ascii_letters = ascii_lowercase + ascii_uppercase
```

`string.`**`digits`**

Gồm các ký tự số từ ‘0’ đến ‘9’

```
digits = '0123456789'
```

`string.`**`octdigits`**

Gồm các ký tự để biểu diễn số dạng bát phân

```
digits = '01234567'
```

`string.`**`hexdigits`**

Gồm các ký tự để biểu diễn số dạng thập lục phân

```
digits = '0123456789abcdefABCDEF'
```

`string.`**`punctuation`**

Gồm các ký tự chấm câu như bên dưới:

```
punctuation = r"""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
```

`string.`**`whitespace`**

Gồm các ký tự được xem như khoảng trắng:

```
whitespace = ' \t\n\r\v\f'
```

`string.`**`printable`**

Gồm các ký tự có thể in được:

```
printable = digits + ascii_letters + punctuation + whitespace
```

Định dạng chuỗi

Lớp string cung cấp khả năng thay thế các biến phức tạp và định dạng giá trị chuỗi thông qua phương thức format().

Phương thức format() sẽ thay thế lần lượt các ký hiệu {} trong chuỗi thành các tham số được truyền vào hàm format.

Xét câu lệnh sau:

```
'{}, {}, {}'.format('a', 'b', 'c')
```

Kết quả xuất ra màn hình sẽ là:

```
'a, b, c'
```

Như ta thấy, phương thức format() đã thay lần lượt các ký hiệu {} trong chuỗi thành các tham số được truyền vào hàm format, cụ thể các tham số ở đây là ‘a’ , ‘b’ và ‘c’.

Ta có thể đánh số cho các ký hiệu {} để chỉ định rõ vị trí {} đó cần điền giá trị của tham số nào:

```
'{2}, {1}, {0}'.format('a', 'b', 'c')
```

Với cú pháp trên,

{2} sẽ được thay thế bằng tham số thứ 2, tức là ‘c’

{1} sẽ được thay thế bằng tham số thứ 1, tức là ‘b’

{0} sẽ được thay thế bằng tham số thứ 0, tức là ‘a’

Vậy kết quả xuất ra màn hình có vẫn có dạng:

```
'c, b, a'
```

Tham số có thể lặp lại nhiều lần trong chuỗi:

```
'{0}{1}{0}'.format('abra', 'cad')
```

Kết quả xuất ra sẽ là:

```
'abracadabra'
```

Chỉ định tham số bằng tên:

```
'Coordinates:  {latitude},  {longitude}'.format(latitude='37.24N',
longitude='-115.81W')
```

Với lệnh trên, {latitude} trong chuỗi sẽ được thay bằng giá trị tham số có tên là latitude, {longitude} trong chuỗi sẽ được thay bằng giá trị tham số có tên là longitude. Kết quả xuất ra:

```
'Coordinates: 37.24N, -115.81W'
```

Phần mở rộng:

Định dạng cho số phức: ta dùng .real để lấy phần thực và .imag để lấy phần ảo

```
>>> c = 3-5j
>>> ('The complex number {0} is formed from the real part {0.real} '
...  'and the imaginary part {0.imag}.').format(c)
'The complex number (3-5j) is formed from the real part 3.0 and the
imaginary part -5.0.'
```

Định dạng phương thức chuyển class thành str:

```
>>> class Point:
...     def __init__(self, x, y):
...         self.x, self.y = x, y
...     def __str__(self):
...         return 'Point({self.x}, {self.y})'.format(self=self)
...
>>> str(Point(4, 2))
'Point(4, 2)'
```

Một số Chỉ định định dạng chuẩn:

Chỉ định chiều dài dữ liệu xuất

Nếu chiều dài dữ liệu xuất nhỏ hơn chiều dài ta chỉ định thì sẽ tự động thêm khoảng trắng vào kết quả.

Option	Meaning
`'<'`	Căn trái nội dung xuất ra. Khoảng trắng được thêm vào (nếu có) nằm ở bên phải nội dung.
`'>'`	Căn phải nội dung xuất ra. Khoảng trắng được thêm vào (nếu có) nằm ở bên trái nội dung.
`'='`	Chỉ sử dụng cho kiểu số. Khoảng trắng thêm vào đặt ở sau dấu của số và trước phần chữ số. Được sử dụng để xuất ra số có dạng như: '+000000120'
`'^'`	Căn giữa.

Ví dụ:

```
'{:<30}'.format('left aligned')
```

Thêm khoảng trắng vào bên phải chuỗi 'left aligned' cho đến khi chuỗi kết quả đủ 30 ký tự. Ở đây, chuỗi 'left aligned' dài 12 ký tự, vậy sẽ thêm vào 18 ký tự khoảng trắng. Kết quả:

```
'left aligned                  '
```

Tương tự ta sẽ có căn phải và căn giữa:

```
>>> '{:>30}'.format('right aligned')
'                 right aligned'
>>> '{:^30}'.format('centered')
'           centered           '
```

Định dạng dấu của số và kiểu dữ liệu

Option	Meaning
`'+'`	Luôn hiển thị dấu của số dù là âm hay dương
`'-'`	Chỉ hiện thị dấu nếu là số âm. Đây là định dạng mặc định.

Type	Meaning
`'b'`	Kiểu nhị phân
`'c'`	Kiểu ký tự.
`'d'`	Số nguyên thập phân
`'o'`	Số bát phân
`'x'`	Thập lục phân, phần chữ cái ký hiệu thường: “abcdef”
`'X'`	Thập lục phân, phần chữ cái ký hiệu in hoa: “ABCDEF”
`'e'`	Số mũ
`'f'`	Số thực

Ví dụ:

```
>>> '{:+f}; {:+f}'.format(3.14, -3.14)  # show it always
'+3.140000; -3.140000'
>>> '{:-f}; {:-f}'.format(3.14, -3.14)  # show only the minus -- same
as '{:f}; {:f}'
'3.140000; -3.140000'
>>> # format also supports binary numbers
>>> "int: {0:d};  hex: {0:x};  oct: {0:o};  bin: {0:b}".format(42)
'int: 42;  hex: 2a;  oct: 52;  bin: 101010'
>>> # with 0x, 0o, or 0b as prefix:
>>> "int: {0:d};  hex: {0:#x};  oct: {0:#o};  bin: {0:#b}".format(42)
'int: 42;  hex: 0x2a;  oct: 0o52;  bin: 0b101010'
```