

Python: nova formatação usando `str.format()` e `format()`

A partir do Python 2.6 existe uma nova notação para formatação de strings, números e outros objetos, substituindo a formatação com o operador `%`. Esta notação é usada principalmente em dois contextos:

contextos de uso	exemplos de uso
<code>str.format(*args, **kwargs)</code> Método aplicado sobre uma string com marcas de substituição <code>{a!b:c}</code> onde: a seleciona o argumento a formatar; b especifica uma conversão e c especifica o formato de apresentação	<pre>>>> fmt = 'valor de {0} com 4 casas: {0:.4f}' >>> fmt.format(math.pi) 'valor de 3.14159265359 com 4 casas: 3.1416' >>> fmt2 = 'valor de {0} com {n:02} casas: {0:.{n}f}' >>> fmt2.format(math.pi, n=5) 'valor de 3.14159265359 com 05 casas: 3.14159'</pre>
<code>format(valor, c)</code> função <i>built-in</i> onde c especifica o formato de apresentação	<pre>>>> format(math.pi, '6.3f') ' 3.142'</pre>

Marcas de substituição

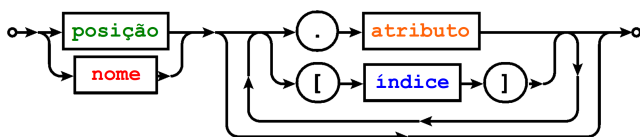


argumento Indica qual argumento do método `format` será apresentado no lugar desta marca de substituição. Veja a seção **Seletor do argumento** abaixo.

conversão Use para forçar a conversão do argumento para string usando as funções `str` ou `repr`. Por padrão, a conversão é feita pela invocação do método `obj.__format__(fmt)`, onde `obj` é o argumento e `fmt` é uma string de formatação (ver abaixo).

formato Especificação do formato de apresentação.

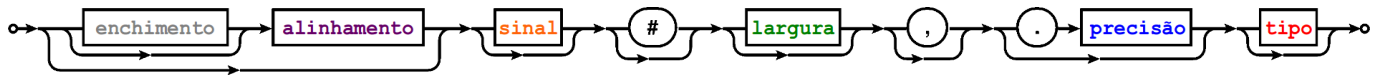
Seletor do argumento



Exemplos

```
>>> format('Diretor', '<.12')
'Diretor.....'
>>> format(3.14159265, '>+8.3f')
'+3.142'
>>> format(123, '0= 6x')
' 0007b'
>>> format(123, '#06x')
'0x007b'
>>> format(12345678.9876, '18.10n')
'      12345678.99'
>>> from locale import setlocale
>>> setlocale(locale.LC_NUMERIC,
...          'de_DE.UTF-8')
'de_DE.UTF-8'
>>> format(12345678.9876, '18.10n')
'      12.345.678,99'
>>>
```

② Especificação do formato



alinhamento Um dos sinais <, ^, > ou =, indicando:

- < alinhamento à esquerda
- ^ centralizado
- > à direita

= à direita com preenchimento após o sinal

O sinal pode ser precedido de um caractere qualquer (exceto { ou }) a ser usado em vez do espaço para preencher o campo se **largura** for definida.

sinal Os caracteres +, - ou _ (um espaço em branco).

- + sempre exibir sinal + ou - à esquerda
- exibir apenas sinal - nos números de negativos
- _ (espaço em branco) exibir sinal - à esquerda de números negativos e branco à esquerda dos positivos.

Use para exibir **0b**, **0o** ou **0x** à esquerda do número nas apresentações de **tipo** binário, octal ou hexadecimal.

largura Número de caracteres da largura total mínima do campo. O conteúdo não é truncado se exceder essa largura. Se o conteúdo for menor, haverá preenchimento conforme o **alinhamento** definido. Se a largura começar com um 0 (zero), o campo será preenchido com zeros à esquerda (igual a **alinhamento 0=**)

, Exibir , (vírgula) como separador de milhares. Para outros separadores de milhares, use o **tipo n**.

precisão Um . (ponto) seguido de um inteiro cuja função depende do **tipo** especificado.

No **tipo s**, precisão é o máximo de caracteres

No **tipo f**, é o número de dígitos após o ponto

No **tipo g** ou **n**, é o total de dígitos significativos

Não pode ser usado com os tipos **b**, **c**, **d** ou **o**

tipo Um dos caracteres abaixo. Se omitido, vale o assinalado com * para cada tipo (ex. **d** para **int**):

- s** **str/unicode** *
- b** **int** como binário
- c** **int** como caractere Unicode correspondente
- d** **int** como decimal *
- o** **int** como octal
- x X** **int** como hexadecimal: **x** caixa baixa, **X** alta
- e E** **float** em notação exponencial:
 - e** caixa baixa, **E** alta
- f F** **float** sem usar notação exponencial
- g G** **float** como nos tipos **e E** ou **f F**, conforme a **precisão** e o expoente *
- n** **float** como no tipo **g**, usando separadores decimal e de milhares conforme o locale ativo
- %** **float** como porcentagem, usando formato do tipo **f**, com o valor $\times 100$, seguido do sinal %

Exemplos

```
>>> format('Diretor','.<12')
'Diretor.....'
>>> format(3.14159265, '_>+8.3f')
'_+3.142'
>>> format(123, '0= 6x')
' 0007b'
>>> format(123, '#06x')
'0x007b'
>>> format(12345678.9876, '18.10n')
'      12345678.99'
>>> from locale import setlocale
>>> setlocale(locale.LC_NUMERIC,
...          'de_DE.UTF-8')
'de_DE.UTF-8'
>>> format(12345678.9876, '18.10n')
'      12.345.678,99'
>>>
```