Formatação avançada em Python

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:

contexto de uso	exemplo de uso
Função *built-in* **`format(valor, ❶)`** onde ❶ é uma especificação de formato.	`>>> format(math.pi, '6.3f')` `' 3.142'`

Método **`str.format(*args, **kwargs)`**, aplicado sobre uma string marcada com **`{❷:❶}`** onde:
❷ é uma especificação de valor e
❶ é uma especificação de formato.
	`>>> 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'`

❶ Especificação de formato

Sintaxe (todos os elementos são opcionais):

`«alinhamento»«sinal»#«largura»,«.precisao»«tipo»`

Elementos do formato

`«alinhamento»`	Um dos caracteres **`<`**, **`^`**, **`>`** ou **`=`**, indicando: **`<`** alinhamento à esquerda **`^`** centralizado **`>`** à direita **`=`** à direita com preenchimento após o sinal O sinal de alinhamento pode ser precedido de um caractere qualquer (exceto **`{`** ou **`}`**) a ser usado em vez do espaço para preencher o campo se **`«largura»`** for definida. Ver ex. 1, 2 e 3.
`«sinal»`	Os caracteres **`+`**, **`-`** ou **`␣`** (um espaço em branco). Ver ex. 2 e 3. **`+`** exibir sinal **`+`** ou **`–`** à esquerda do número **`-`** exibir apenas sinal **`–`** à esquerda de números negativos **`␣`** (espaço em branco) exibir sinal **`–`** à esquerda de números negativos ou espaço em 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 obter outros separadores de milhares, use o **`«tipo» n`**.
`«.precisao»`	Um **`.`** (ponto) seguido de um inteiro cuja função depende do **`«tipo»`** especificado. No **`«tipo» s`**, precisão é o número máximo de caracteres a exibir. No **`«tipo» f`**, é o número de casas decimais após o ponto. No **`«tipo» g`** ou **`n`**, é o número total de algarismos significativos.
`«tipo»`	Um dos caracteres abaixo (ver tabela de ***Tipos de formato***): **`s`** para apresentação de strings **`b c d o x X`** para apresentação de números inteiros **`e E f F g G n %`** para números de ponto flutuante

Tipos de formato

s	Exibir string.
b	Exibir inteiro como número binário.
c	Exibir inteiro como caractere Unicode correspondente.
d	Exibir inteiro como número decimal.
o	Exibir inteiro como número octal.
x X	Exibir inteiro como número hexadecimal (letras em caixa baixa ou alta).
e E	Exibir número de ponto flutuante em notação exponencial (indicador do expoente em caixa baixa ou alta).
f F	Exibir número de ponto flutuante sem usar notação exponencial.
g G	Exibir número de ponto flutuante como nos tipos **e E** ou **f F**, dependendo da precisão e do expoente.
n	Exibir número de ponto flutuante como no tipo **g**, mas usando separadores decimal e de milhares conforme a configuração de **locale** ativa.
%	Exibir número de ponto flutuante como porcentagem, usando formato do tipo **f**, com o valor multiplicado por 100 e seguido do sinal **%**.

❷ Especificação de valor

Exemplos

	expressão	resultado	equivalente usando %
1	`format('Diretor','.<12')`	`'Diretor.....'`	
2	`format(math.pi, '_>+8.3f')`	`'__+3.142'`	
3	`format(123, '0= 6x')`	`' 007b'`	
	`format(123, '#06x')`	`'0x007b'`	
	`format(12345678.9876,'18.10n')`	`'   12.345.678,9876'`	