Python String format() Method

Axlesoft CPD
www.axlesoft.com

Formatting

The format() method uses a mini-language to control how values are displayed in string form

Basics

```
>>> s = '{} is measured in {}'
>>> s.format('Length', 'metres')
'Length is measured in metres'
```

Each {} consumes the next parameter

```
>>>'{} + {} = {}'.format(2, 2, 4)
'2 + 2 = 4'
```

Can call format() on a literal string

Alignment

```
>>> '{:8}'.format('wide')
'wide    '
```

Minimum field width

```
>>>'{:<8}|{:^8}|{:>8}'.format(1, 2, 3)
'1       |   2    |       3'
```

Align left, centre or right within field

```
>>>'{:~<8}|{:~^8}|{:~>8}'.format(1, 2, 3)
'1~~~~~~~|~~~2~~~~|~~~~~~~3'
```

Use '~' as fill character

Integers

```
>>> '{:d}'.format(23)
'23'
```

Show as integer

```
>>>'{:b} {:o} {:x}'.format(165, 165, 165)
'10100101 245 a5'
```

Different bases - binary, octal, hexadecimal

```
>>>'{:X}'.format(165)
'A5'
```

Capitalise letters in hexadecimal

```
>>>'{:#b} {:#o} {:#x}'.format(165, 165, 165)
'0b10100101 0o245 0xa5'
```

Include 0b, 0o, 0x prefix

```
>>>'{:*>8d}'.format(23)
'******23'
```

All the above work with alignment

Floats

```
>>> '{:f}'.format(23.1)
'23.100000'
```

By default shows 6 decimal places

```
>>> '{:.2f}'.format(23.1)
'23.10'
```

Sets the number of decimal places

```
>>> '{:6.2f}'.format(23.1)
' 23.10'
```

Sets the field width, 6 characters including the point

```
>>> '{:e}'.format(23.1)
'2.310000e+01'
```

Uses scientific notation rather than fixed point

Numbers (General)

```
>>> '{:08.2f}'.format(23.1)
'00023.10'
>>> '{:08.2f}'.format(-23.1)
'-0023.10'
```

Using 08 rather than 8 pads the field with zeros. The zeros take account of sign. This also works with integers

```
>>>'{:+d} {:+d}'.format(15, -15)
'+15 -15'
```

+ sign means positive and negative values are signed

```
>>>'{: d} {: d}'.format(15, -15)
' 15 -15'
```

Space means positive values have a space instead of a + sign

```
>>>'{:,d}'.format(65536)
'65,536'
```

Use a thousands separator

Combinations

The features above can be combined in various ways. The syntax requires all features to be placed in the correct order, for example {:08.2f} is valid, {:08f.2} is not.

```
>>>'{:+12,.2f}'.format(15634.999)
'  +15,635.00'
```

Float, precision 2, width 12, always with sign, thousands indicators

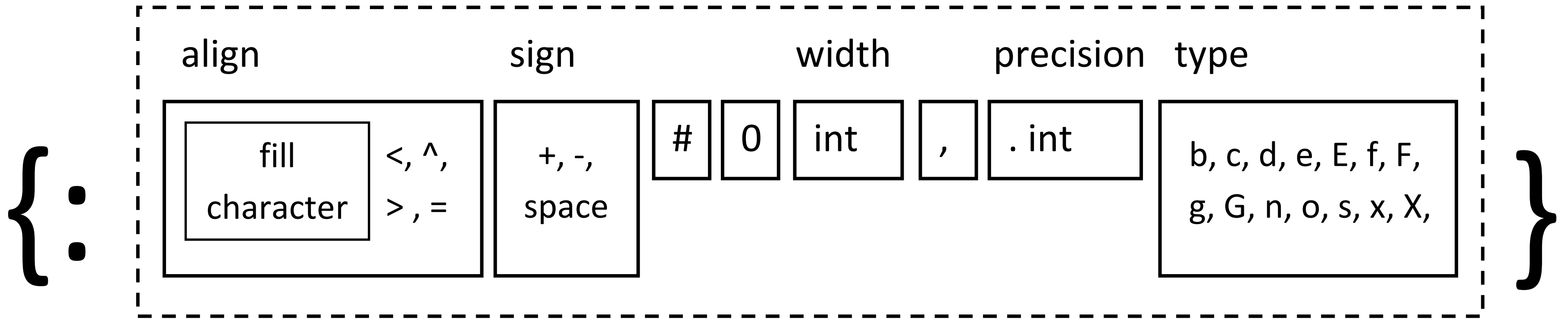
```
>>>'{:08x}'.format(65535)
'0000ffff'
```

8 digit, zero padded hex

```
>>>'{:#010x}'.format(65535)
'0x0000ffff'
```

8 digit, zero padded hex, with 0x prefix. Total width is 10

These are the most commonly used features of string formatting. See the documentation at python.org for a full descripion.



All formatting elements are optional.
Any elements which are used *must* appear in the order above

Align	Optional fill character, followed by <, ^, > or =
Sign	+, - or space character to indicate is +ve numbers should have a +, a preceding space, or nothing
#	# character indicates "alternate form", eg 0x prefix for hexadecimal
0	0 (zero) character indicates that numerical values should use 0 for padding
width	Integer, indicates *minimum* width of field
,	, character indicates that thousand separators should be used with numerical values
. precision	. character followed by an integer to show how many decimal places should be used with float values
type	Types of data and required representation, eg d for decimal integer, x for hexadecimal etc