7.1.3.2. Format examples

This section contains examples of the new format syntax and comparison with the old %-formatting.

In most of the cases the syntax is similar to the old %-formatting, with the addition of the {} and with : used instead of %. For example, '%03.2f' can be translated to '{:03.2f}'.

The new format syntax also supports new and different options, shown in the follow examples.

Accessing arguments by position:

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

'a, b, c'

**>>>** '{}, {}, {}'.format('a', 'b', 'c') *# 3.1+ only*

'a, b, c'

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

'c, b, a'

**>>>** '{2}, {1}, {0}'.format(\*'abc') *# unpacking argument sequence*

'c, b, a'

**>>>** '{0}{1}{0}'.format('abra', 'cad') *# arguments' indices can be repeated*

'abracadabra'

Accessing arguments by name:

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

'Coordinates: 37.24N, -115.81W'

**>>>** coord = {'latitude': '37.24N', 'longitude': '-115.81W'}

**>>>** 'Coordinates: {latitude}, {longitude}'.format(\*\*coord)

'Coordinates: 37.24N, -115.81W'

Accessing arguments’ attributes:

**>>>** 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.'

**>>> 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)'

Accessing arguments’ items:

**>>>** coord = (3, 5)

**>>>** 'X: {0[0]}; Y: {0[1]}'.format(coord)

'X: 3; Y: 5'

Replacing %s and %r:

**>>>** "repr() shows quotes: {!r}; str() doesn't: {!s}".format('test1', 'test2')

"repr() shows quotes: 'test1'; str() doesn't: test2"

Aligning the text and specifying a width:

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

'left aligned '

**>>>** '{:>30}'.format('right aligned')

' right aligned'

**>>>** '{:^30}'.format('centered')

' centered '

**>>>** '{:\*^30}'.format('centered') *# use '\*' as a fill char*

'\*\*\*\*\*\*\*\*\*\*\*centered\*\*\*\*\*\*\*\*\*\*\*'

Replacing %+f, %-f, and % f and specifying a sign:

**>>>** '{:+f}; {:+f}'.format(3.14, -3.14) *# show it always*

'+3.140000; -3.140000'

**>>>** '{: f}; {: f}'.format(3.14, -3.14) *# show a space for positive numbers*

' 3.140000; -3.140000'

**>>>** '{:-f}; {:-f}'.format(3.14, -3.14) *# show only the minus -- same as '{:f}; {:f}'*

'3.140000; -3.140000'

Replacing %x and %o and converting the value to different bases:

**>>>** *# 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'

Using the comma as a thousands separator:

**>>>** '{:,}'.format(1234567890)

'1,234,567,890'

Expressing a percentage:

**>>>** points = 19

**>>>** total = 22

**>>>** 'Correct answers: {:.2%}.'.format(points/total)

'Correct answers: 86.36%'

Using type-specific formatting:

**>>> import** **datetime**

**>>>** d = datetime.datetime(2010, 7, 4, 12, 15, 58)

**>>>** '{:%Y-%m-%d %H:%M:%S}'.format(d)

'2010-07-04 12:15:58'

Nesting arguments and more complex examples:

**>>> for** align, text **in** zip('<^>', ['left', 'center', 'right']):

**...**  '{0:{fill}{align}16}'.format(text, fill=align, align=align)

**...**

'left<<<<<<<<<<<<'

'^^^^^center^^^^^'

'>>>>>>>>>>>right'

>>>

**>>>** octets = [192, 168, 0, 1]

**>>>** '{:02X}{:02X}{:02X}{:02X}'.format(\*octets)

'C0A80001'

**>>>** int(\_, 16)

3232235521

>>>

**>>>** width = 5

**>>> for** num **in** range(5,12):

**...**  **for** base **in** 'dXob':

**...**  print('{0:{width}{base}}'.format(num, base=base, width=width), end=' ')

**...**  print()

**...**

5 5 5 101

6 6 6 110

7 7 7 111

8 8 10 1000

9 9 11 1001

10 A 12 1010

11 B 13 1011