# Formatting Numbers in Python.

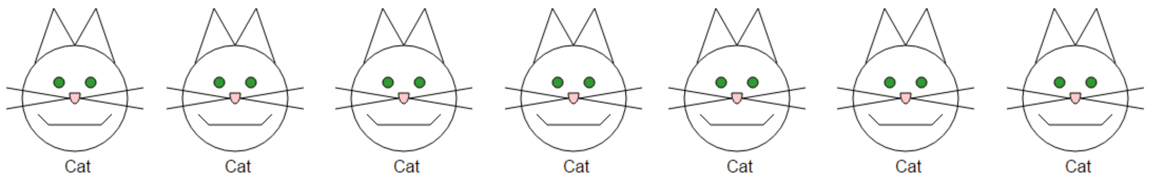


**Figure 1:** I have 7 cats.

The table below shows us different ways that we may format a number in Python. In this instance, I have chosen the number, 7.

| Syntax: | Output: |
|---|---|
| print("I have {0:d} cats".format(7,6,5,4)) | I have 7 cats |
| print("I have {0:3d} cats".format(7,6,5,4)) | I have   7 cats |
| print("I have {0:03d} cats".format(7,6,5,4)) | I have 007 cats |
| print("I have {0:f} cats".format(7,6,5,4)) | I have 7.000000 cats |
| print("I have {0:.2f} cats".format(7,6,5,4)) | I have 7.00 cats |

I will take every entry of the above table, individually, and shall explain what is going on.

1. **print ("I have {0:d} cats".format(7,6,5,4))**

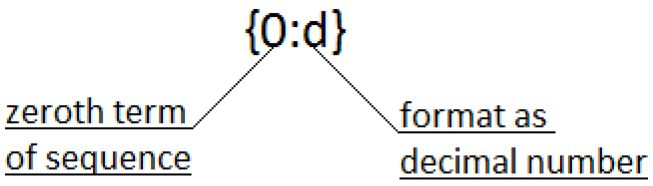


**Figure 2:** The contents of the chain parenthesis analysed.

In the above command, we specify, to python, that we wish to format the zeroth number-element in the listed sequence:

(7,6,5,4)

.

This is what the:

0

part of:

{0:d}

is for.

In this instance the zeroth[1] number-element in the listed sequence is:

7

.

Therefore, it will be the number, 7, that will be formatted and printed by Python.

We use a

d

in the chain parenthesis, to let Python know that we wish to format the number:

7

as an ordinary decimal number.

When we give the command:

```
>>> print("I have {0:d} cats".format(7,6,5,4))
```

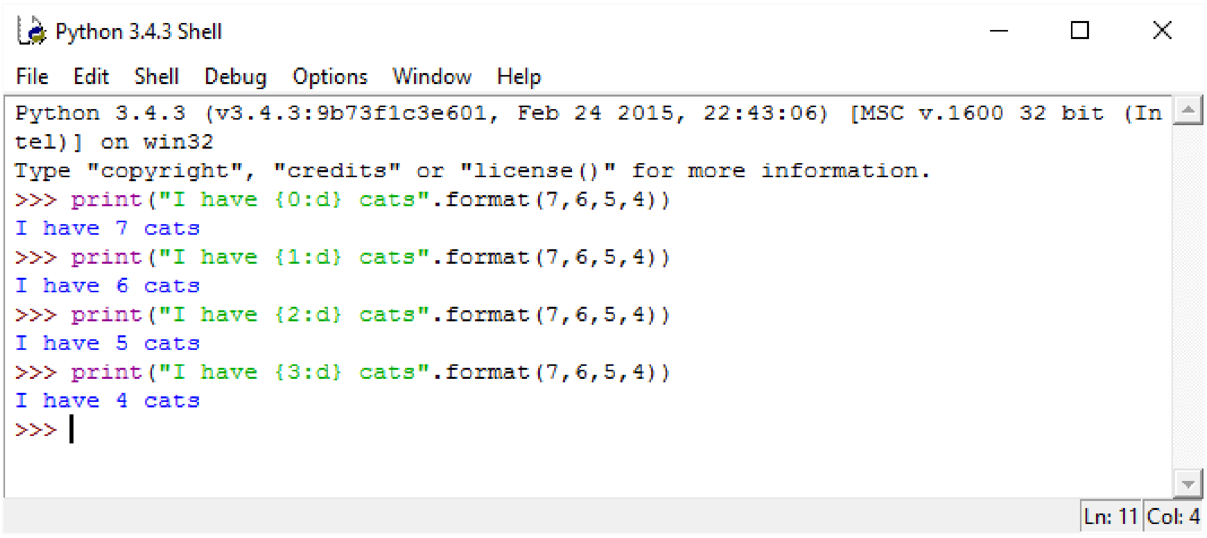
to Python, Python outputs:

```
I have 7 cats
```

.

Below are examples of what occurs when we give formatting commands such as these to a Python Interactive Window:

[1] In programing, it is conventional to begin counting beginning at 0, not beginning at 1. Therefore, zeroth, or 0th, is an ordinal number. Hence: Zeroth, First, second … Hence: 0th, 1st, 2nd …



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("I have {0:d} cats".format(7,6,5,4))
I have 7 cats
>>> print("I have {1:d} cats".format(7,6,5,4))
I have 6 cats
>>> print("I have {2:d} cats".format(7,6,5,4))
I have 5 cats
>>> print("I have {3:d} cats".format(7,6,5,4))
I have 4 cats
>>> 
```

**Figure 3:** In the above example, we, systematically, format all of the number-elements in the sequence: (7,6,5,4). We do this by altering the value of the number before the colon in the chain parenthesis.

## 2. print ("I have {0:3d} cats".format(7,6,5,4))

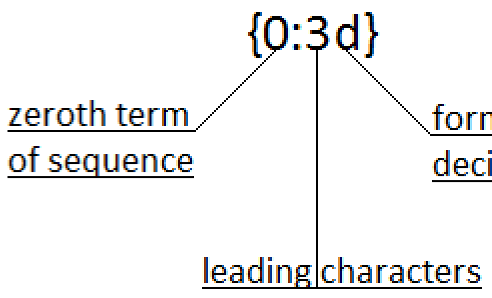


**Figure 4:** The contents of the chain parenthesis analysed.

In the above command, we specify, to python, that we wish to format the zeroth number-element in the listed sequence:

(7,6,5,4)

.

This is what the

0

part of:

{0:3d}

is for.

In this instance the zeroth number-element in the listed sequence is:

7

.

Therefore, it will be the number, 7, that will be formatted and printed by Python.

We use a

d

in the chain parenthesis, to let Python know that we wish to format the number:

7

as an ordinary decimal number.

The

3

character tells python that we wish the decimal number, i.e. 7, to be the third character after leading[2] characters.  As we do not specify what form that we wish for these leading characters to take, then:

7

will be the third character after two leading spaces.

[2] In Mathematics, the two zeros that precede the number, 7, in a number such as: 007 , are termed ‘leading zeros.’  In Mathematics, the two zeros that *follow* a number such as:  0.700 , are termed ‘trailing zeros.’

When we give the command:

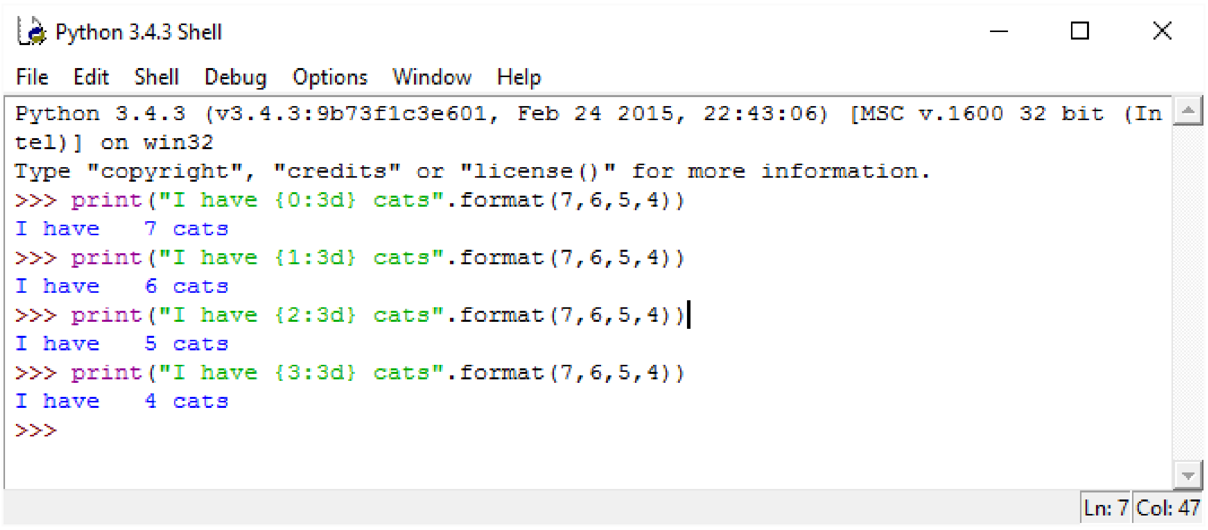
>>> print("I have {0:3d} cats".format(7,6,5,4))

to Python, Python outputs:

I have   7 cats

.

Below are examples of what occurs when we give formatting commands such as these to a Python Interactive Window:



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("I have {0:3d} cats".format(7,6,5,4))
I have   7 cats
>>> print("I have {1:3d} cats".format(7,6,5,4))
I have   6 cats
>>> print("I have {2:3d} cats".format(7,6,5,4))
I have   5 cats
>>> print("I have {3:3d} cats".format(7,6,5,4))
I have   4 cats
>>>
```

**Figure 5:** In the above example, we, systematically, format all of the number-elements in the sequence: (7,6,5,4). We do this by altering the value of the number before the colon in the chain parenthesis.

3. **<u>print ("I have {0:03d} cats".format(7,6,5,4))</u>**

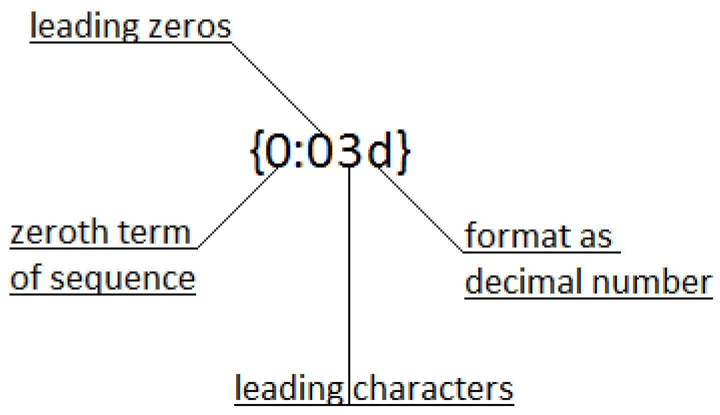


**Figure 6:** The contents of the chain parenthesis analysed.

In the above command, we specify, to python, that we wish to format the zeroth number-element in the listed sequence:

(7,6,5,4)

.

This is what the

0

part of:

{0:03d}

is for.

In this instance the zeroth number-element in the listed sequence is:

7

.

Therefore, it will be the number, 7, that will be formatted and printed by Python.

We use a

d

in the chain parenthesis, to let Python know that we wish to format the number:

7

as an ordinary decimal number.

The

3

character tells python that we wish the decimal number, i.e. 7, to be the third character after leading[3] characters.

The:

0

prior to the:

3

and following the:

:

in:

{0:03d}

signifies the leading character:

zero

.

Therefore:

7

will be the third character after two leading zeros.

[3] In Mathematics, the two zeros that precede the number, 7, in a number such as: 007 , are termed ‘leading zeros.’ In Mathematics, the two zeros that *follow* a number such as: 0.700 , are termed ‘trailing zeros.’

When we give the command:

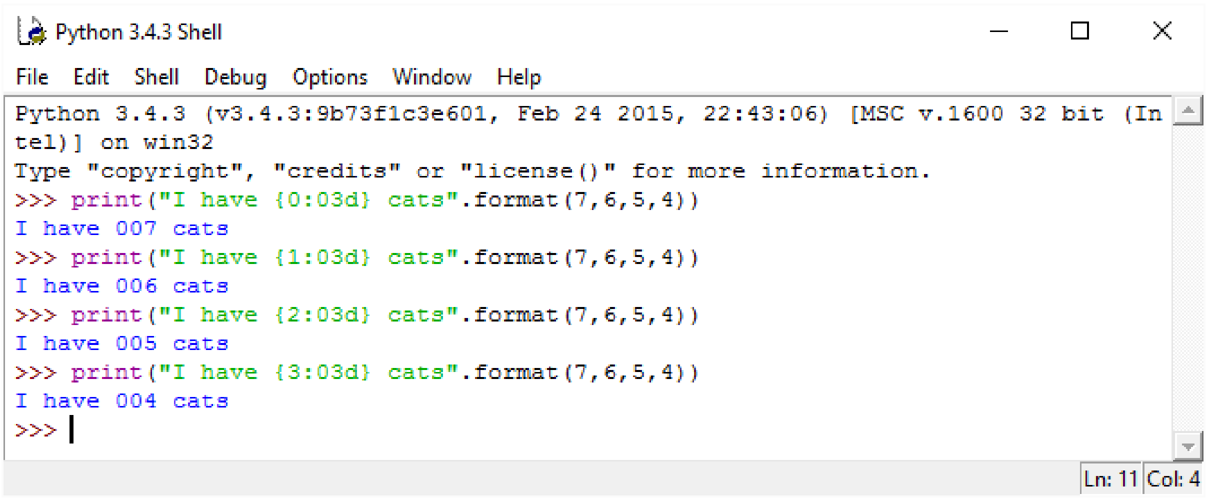
>>> print("I have {0:03d} cats".format(7,6,5,4))

to Python, Python outputs:

I have 007 cats

.

Below are examples of what occurs when we give formatting commands such as these to a Python Interactive Window:



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("I have {0:03d} cats".format(7,6,5,4))
I have 007 cats
>>> print("I have {1:03d} cats".format(7,6,5,4))
I have 006 cats
>>> print("I have {2:03d} cats".format(7,6,5,4))
I have 005 cats
>>> print("I have {3:03d} cats".format(7,6,5,4))
I have 004 cats
>>> 
```

**Figure 7:** In the above example, we, systematically, format all of the number-elements in the sequence: (7,6,5,4). We do this by altering the value of the number before the colon in the chain parenthesis.

**4. <u>print ("I have {0:f} cats".format(7,6,5,4))</u>**

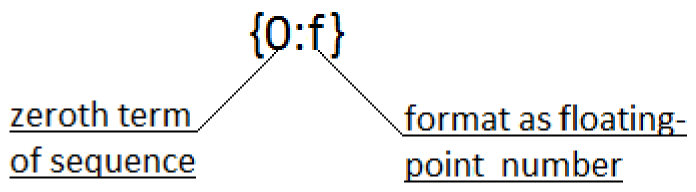


**Figure 8:** The contents of the chain parenthesis analysed.

In the above command, we specify, to python, that we wish to format the zeroth number-element in the listed sequence:

(7,6,5,4)

.

This is what the

0

part of:

{0:f}

is for.

In this instance the zeroth number-element in the listed sequence is:

7

.

Therefore, it will be the number, 7, that will be formatted and printed by Python.

We use an:

f

in the chain parenthesis, to let Python know that we wish to format the number:

7

as a floating-point number[4].

When we give the command:

```
>>> print("I have {0:f} cats".format(7,6,5,4))
```

to Python, Python outputs:

```
I have 7.000000 cats
```

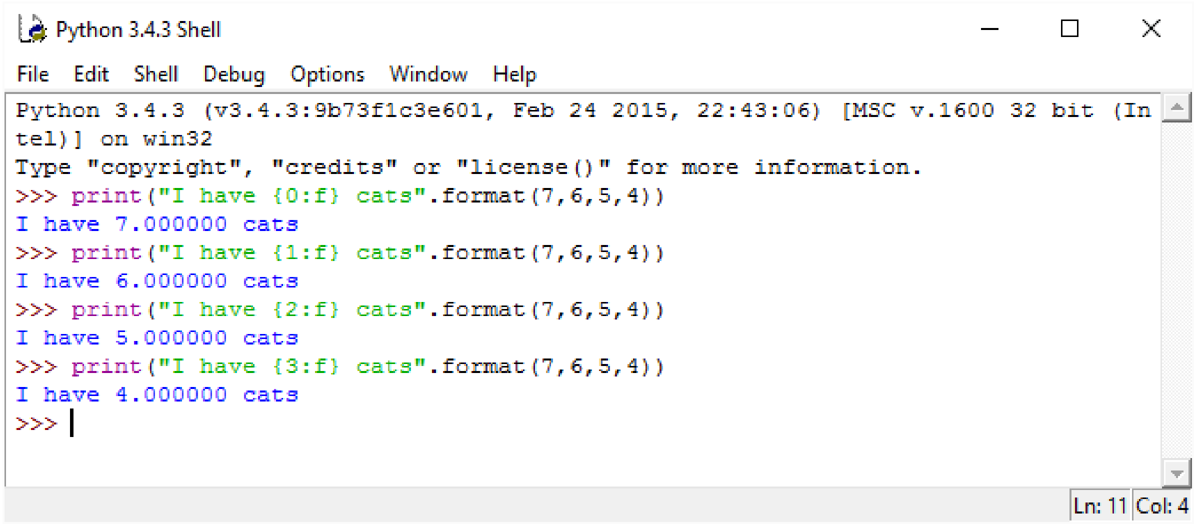
.

As we can see, the number,

7,

its being a *float* is followed by a decimal point and six trailing zeros.

Below are examples of what occurs when we give formatting commands such as these to a Python Interactive Window:



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("I have {0:f} cats".format(7,6,5,4))
I have 7.000000 cats
>>> print("I have {1:f} cats".format(7,6,5,4))
I have 6.000000 cats
>>> print("I have {2:f} cats".format(7,6,5,4))
I have 5.000000 cats
>>> print("I have {3:f} cats".format(7,6,5,4))
I have 4.000000 cats
>>> |
```

**Figure 9:** In the above example, we, systematically, format all of the number-elements in the sequence: (7,6,5,4). We do this by altering the value of the number before the colon in the chain parenthesis.

[4] In programming, this is generally termed: ‘float.’

## 5. print ("I have {0:.2f} cats".format(7,6,5,4))

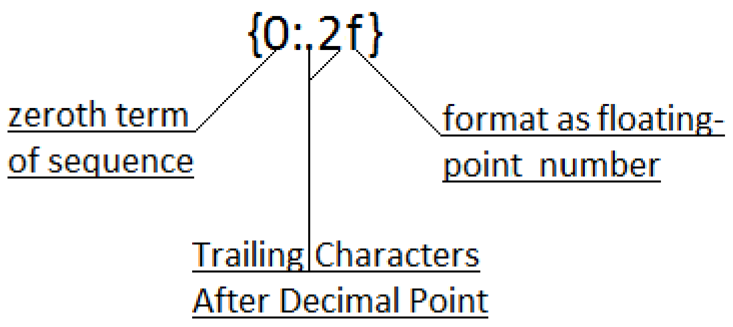

{0:.2f}
zeroth term of sequence
format as floating-point number
Trailing Characters After Decimal Point

**Figure 10:** The contents of the chain parenthesis analysed.

In the above command, we specify, to python, that we wish to format the zeroth number-element in the listed sequence:

(7,6,5,4)

.

This is what the

0

part of:

{0:.2f}

is for.

In this instance the zeroth number-element in the listed sequence is:

7

.

Therefore, it will be the number, 7, that will be formatted and printed by Python.

We use a

f

in the chain parenthesis, to let Python know that we wish to format the number:

7

as a float.

The

.2

characters tell python that we wish the floating-point number, i.e. 7, to be followed, after a decimal point, by two trailing characters, in this instance, zeros.

When we give the command:

>>> print("I have {0:.2f} cats".format(7,6,5,4))

to Python, Python outputs:

I have 7.00 cats

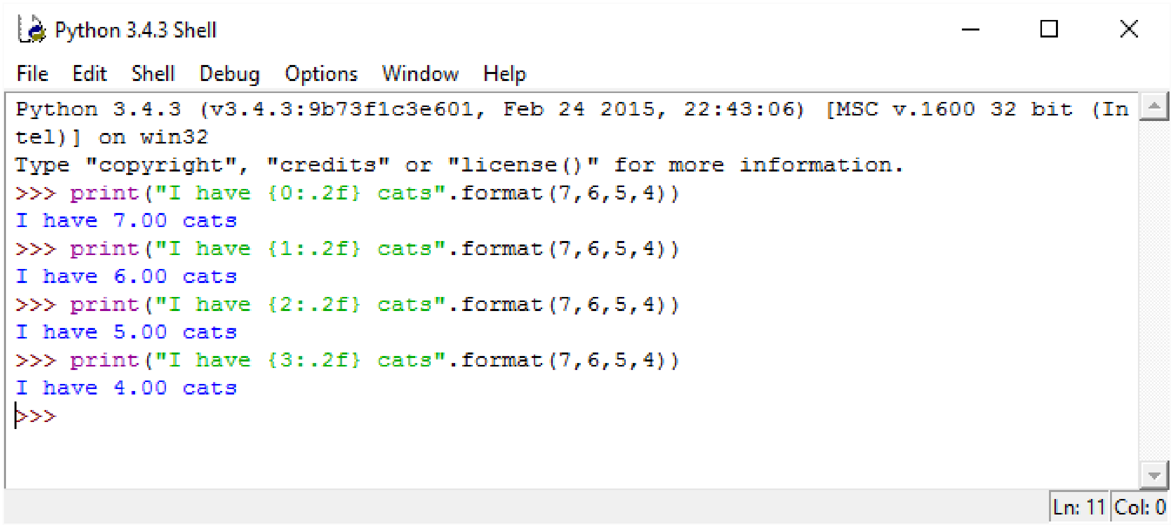
.

As we can see, from the above example, the number:

7

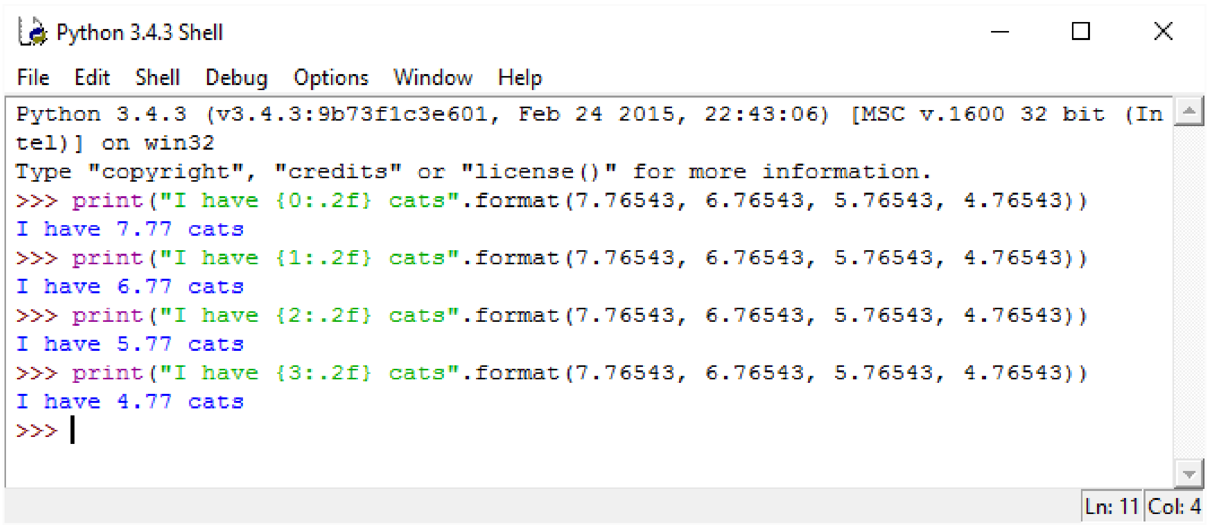
, is now followed by a decimal point and two trailing zeros, as per our command.

Below are examples of what occurs when we give formatting commands such as these to a Python Interactive Window:



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("I have {0:.2f} cats".format(7,6,5,4))
I have 7.00 cats
>>> print("I have {1:.2f} cats".format(7,6,5,4))
I have 6.00 cats
>>> print("I have {2:.2f} cats".format(7,6,5,4))
I have 5.00 cats
>>> print("I have {3:.2f} cats".format(7,6,5,4))
I have 4.00 cats
>>>
```

**Figure 11:** In the above example, we, systematically, format all of the number-elements in the sequence: (7,6,5,4). We do this by altering the value of the number before the colon in the chain parenthesis.



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("I have {0:.2f} cats".format(7.76543, 6.76543, 5.76543, 4.76543))
I have 7.77 cats
>>> print("I have {1:.2f} cats".format(7.76543, 6.76543, 5.76543, 4.76543))
I have 6.77 cats
>>> print("I have {2:.2f} cats".format(7.76543, 6.76543, 5.76543, 4.76543))
I have 5.77 cats
>>> print("I have {3:.2f} cats".format(7.76543, 6.76543, 5.76543, 4.76543))
I have 4.77 cats
>>> 
```

**Figure 12:** In this instance, the characters that trail after the decimal point are *significant*, i.e. not zero. Python rounds up 7.76543 to 7.77.