Python Review: String Formatting

Examples of String Formatting

Description	Example
Insert values into a string.	`"{} and {}".format('salt', 'pepper')`
Use positional arguments.	`"{0} {1} {0} {1}".format('a', 'b')`
Use leading zeroes with an integer.	`"Zeroes, {:05d}".format(n)`
Display sign on an integer.	`"Positive 42 is {:+d}".format(42)`
Round floats, to two decimals.	`"Pi is {:.2f}".format(3.14159)`
Centre and right-align text within a fixed width.	`"{:^10} and {:>10}".format('hi', 'bye')`
Pad formatted text with a specified character.	`"{:*^20}".format('stars')`
Set width based on a variable, *n*.	`"{:>{width}}".format('yes',width=n)`

* See the official Format Specification Mini-Language for more examples.

Write programs that accomplish each task. Use proper conventions for variable names, input prompts, output statements, and program structure. Do not assume that the user will enter the correct data type.

1. Read a positive integer, *n*, from the user and display a table of n^2 and n^3 from 1-*n*. Right-align each column. For example, when *n* = 3 the following table is made.

```
N    n^2   n^3
1      1     1
2      4     8
3      9    27
```

2. Read a sequence of positive integers less than 1 000 from the user, and display them in descending order as zero-padded three-digit integers (e.g. 008).

3. Read an arbitrary number of prices, then generate a bill of sale, including subtotal, 13% tax, and final cost, all displayed as right-aligned currency (two decimal) values.

4. Read an arbitrary number of number of positive integers from the user, then display them in a centred column, padded by dashes, where the width of the column corresponds to the number of digits in the largest number.

5. Rewrite the code below so that it works for a tuple of arbitrary length. Note that the asterisk before `t` in the format method unpacks the tuple.

```
t = (2, 4, 7)
"the 3 numbers are {}, {} and {}".format(*t)
```

Hint: how can you build up a format string before applying `format` to it?